

University of Warwick institutional repository: <http://go.warwick.ac.uk/wrap>

A Thesis Submitted for the Degree of PhD at the University of Warwick

<http://go.warwick.ac.uk/wrap/60439>

This thesis is made available online and is protected by original copyright.

Please scroll down to view the document itself.

Please refer to the repository record for this item for information to help you to cite it. Our policy information is available from the repository home page.

Library Declaration and Deposit Agreement

1. STUDENT DETAILS

Please complete the following:

Full name: Thomas James Osgood

University ID number: 0504962

2. THESIS DEPOSIT

2.1 I understand that under my registration at the University, I am required to deposit my thesis with the University in BOTH hard copy and in digital format. The digital version should normally be saved as a single pdf file.

2.2 The hard copy will be housed in the University Library. The digital version will be deposited in the University's Institutional Repository (WRAP). Unless otherwise indicated (see 2.3 below) this will be made openly accessible on the Internet and will be supplied to the British Library to be made available online via its Electronic Theses Online Service (EThOS) service.

[At present, theses submitted for a Master's degree by Research (MA, MSc, LLM, MS or MMedSci) are not being deposited in WRAP and not being made available via EThOS. This may change in future.]

2.3 In exceptional circumstances, the Chair of the Board of Graduate Studies may grant permission for an embargo to be placed on public access to the hard copy thesis for a limited period. It is also possible to apply separately for an embargo on the digital version. (Further information is available in the *Guide to Examinations for Higher Degrees by Research*.)

2.4 If you are depositing a thesis for a Master's degree by Research, please complete section (a) below. For all other research degrees, please complete both sections (a) and (b) below:

(a) Hard Copy

I hereby deposit a hard copy of my thesis in the University Library to be made publicly available to readers ~~(please delete as appropriate)~~ EITHER immediately OR after an embargo period of months/years as agreed by the Chair of the Board of Graduate Studies.

I agree that my thesis may be photocopied. YES / ~~NO~~ (Please delete as appropriate)

(b) Digital Copy

I hereby deposit a digital copy of my thesis to be held in WRAP and made available via EThOS.

Please choose one of the following options:

EITHER My thesis can be made publicly available online. YES / ~~NO~~ (Please delete as appropriate)

~~OR--My thesis can be made publicly available only after.....[date]--(Please give date)~~
YES / NO (Please delete as appropriate)

~~OR--My full thesis cannot be made publicly available online but I am submitting a separately identified--additional, abridged version that can be made available online--~~
YES / NO (Please delete as appropriate)

~~OR--My thesis cannot be made publicly available online--~~ YES / NO (Please delete as appropriate)

3. GRANTING OF NON-EXCLUSIVE RIGHTS

Whether I deposit my Work personally or through an assistant or other agent, I agree to the following:

Rights granted to the University of Warwick and the British Library and the user of the thesis through this agreement are non-exclusive. I retain all rights in the thesis in its present version or future versions. I agree that the institutional repository administrators and the British Library or their agents may, without changing content, digitise and migrate the thesis to any medium or format for the purpose of future preservation and accessibility.

4. DECLARATIONS

(a) I DECLARE THAT:

- I am the author and owner of the copyright in the thesis and/or I have the authority of the authors and owners of the copyright in the thesis to make this agreement. Reproduction of any part of this thesis for teaching or in academic or other forms of publication is subject to the normal limitations on the use of copyrighted materials and to the proper and full acknowledgement of its source.
- The digital version of the thesis I am supplying is the same version as the final, hard-bound copy submitted in completion of my degree, once any minor corrections have been completed.
- I have exercised reasonable care to ensure that the thesis is original, and does not to the best of my knowledge break any UK law or other Intellectual Property Right, or contain any confidential material.
- I understand that, through the medium of the Internet, files will be available to automated agents, and may be searched and copied by, for example, text mining and plagiarism detection software.

(b) IF I HAVE AGREED (in Section 2 above) TO MAKE MY THESIS PUBLICLY AVAILABLE DIGITALLY, I ALSO DECLARE THAT:

- I grant the University of Warwick and the British Library a licence to make available on the Internet the thesis in digitised format through the Institutional Repository and through the British Library via the EThOS service.
- If my thesis does include any substantial subsidiary material owned by third-party copyright holders, I have sought and obtained permission to include it in any version of my thesis available in digital format and that this permission encompasses the rights that I have granted to the University of Warwick and to the British Library.

5. LEGAL INFRINGEMENTS

I understand that neither the University of Warwick nor the British Library have any obligation to take legal action on behalf of myself, or other rights holders, in the event of infringement of intellectual property rights, breach of contract or of any other right, in the thesis.

Please sign this agreement and return it to the Graduate School Office when you submit your thesis.

Student's signature:  Date: August, 2013

THOMAS J. OSGOOD

SEMANTIC LABELLING OF ROAD SCENES USING SUPERVISED AND
UNSUPERVISED MACHINE LEARNING WITH LIDAR-STEREO SENSOR
FUSION

THESIS

Submitted to the University of Warwick

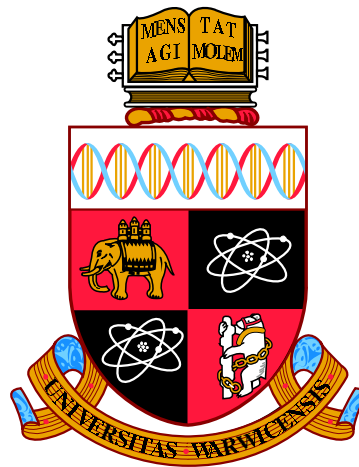
for the degree of

DOCTOR OF PHILOSOPHY

International Manufacturing Centre

SEMANTIC LABELLING OF ROAD SCENES USING
SUPERVISED AND UNSUPERVISED MACHINE LEARNING
WITH LIDAR-STEREO SENSOR FUSION

THOMAS J. OSGOOD



THESIS

Submitted to the University of Warwick

for the degree of

DOCTOR OF PHILOSOPHY

International Manufacturing Centre

August 2013 – VERSION 2.1

THE UNIVERSITY OF
WARWICK

Thomas J. Osgood: *Semantic labelling of road scenes using supervised and unsupervised machine learning with LIDAR-stereo sensor fusion*, Primarily for the improvement of automotive safety and navigation systems, © August 2013

*All of the biggest technological inventions created by man - the air plane, the auto mobile,
the computer - says little about his intelligence, but speaks volumes about his laziness.*

— Mark Kennedy

CONTENTS

Acknowledgements [xvii](#)

Declaration [xviii](#)

Publications [xix](#)

Abstract [xx](#)

Acronyms [xxi](#)

1	INTRODUCTION	1
1.1	Advanced driver assistance systems (ADAS)	1
1.2	Basic concepts	5
1.2.1	Semantic labelling	5
1.2.2	Stereo vision	6
1.2.3	LIDAR	8
1.2.4	Supervised learning	11
1.2.5	Unsupervised learning	12
1.3	Aims and contributions of thesis	12
1.4	Thesis outline	15
2	LITERATURE REVIEW	19
2.1	Depth from stereo	19
2.1.1	Image capture, preparation and pre-processing	20
2.1.2	Feature matching	23
2.1.3	Verification of matching	27
2.1.4	Distance calculation	28
2.2	LIDAR to camera calibration	30

2.3	Machine learning	31
2.3.1	Artificial neural networks	31
2.3.2	Support vector machines	34
2.3.3	K-means clustering for unsupervised learning	36
2.4	Image processing	38
2.4.1	Edge detection	38
2.4.2	Segmentation	40
2.5	General semantic classification	41
3	LIDAR CALIBRATION AND FUSION FOR IMPROVED DEPTH MAPS	47
3.1	Calibration overview	48
3.2	Problem definition	49
3.3	Approach	52
3.3.1	Nelder-Mead algorithm	52
3.3.2	Definition of the objective function	53
3.4	Previous work to determine number of required targets	58
3.5	Collection of test data	61
3.6	Results and discussion	68
3.6.1	Calibration results	68
3.6.2	Validation of results using initial conditions	70
3.6.3	Validation using independent samples	72
3.7	Fusion and merging of sensor data	72
3.7.1	Final calibration	74
3.7.2	LIDAR interpolation	75
3.8	Theoretical stereo noise performance	79
4	DATA COLLECTION AND SET DESCRIPTION	83
4.1	Data collection hardware	84
4.1.1	The ibeo LUX LIDAR	84
4.1.2	Videre STOC stereo on chip system	85

4.1.3	Point Grey Research bumblebee2	87
4.1.4	Computer hardware	89
4.2	Notes on data collection	89
4.3	Data sets	95
4.3.1	The “A46” set	95
4.3.2	The “RX Residential” set	96
4.3.3	The “Berkswell Auto” set	97
4.3.4	The “Berkswell Manual” set	98
4.3.5	Sample images from the sets	99
5	IMAGE SEGMENTATION AND ANALYSIS FOR TRAINING	101
5.1	Colour and contrast correction	102
5.2	Hybrid image segmentation	105
5.2.1	Step 1: Canny edge detection with region closing	107
5.2.2	Step 2: Depth map segmentation	111
5.2.3	Step 3: Mean-shift texture segmentation	114
5.2.4	Step 4: Segmentation fusion	116
5.2.5	Performance of segmentation fusion method	119
5.3	Choice of colour space representation	122
5.4	Texture and shape properties	128
5.5	Choice of depth augmented descriptors	138
5.5.1	Point cloud orientation	140
5.5.2	Point cloud planarity	143
5.6	Discriminatory analysis with principal component analysis	150
5.7	Conclusion	157
6	SUPERVISED LEARNING WITH MANUALLY LABELLED IMAGES	159
6.1	Data preparation	160
6.1.1	Frame selection and manual labelling	160
6.1.2	Labelled dataset description	164

6.2	Experiment setup	168
6.2.1	Support vector machine configuration	169
6.2.2	Neural network configuration	172
6.2.3	Comparison methodology	176
6.3	Training results	177
6.4	Conclusion	187
7	EXPERIMENT WITH UNSUPERVISED LEARNING	191
7.1	Experiment setup	194
7.2	Choosing the optimal number of sub-classes	196
7.3	Classification results	200
7.4	Conclusion	203
8	CONCLUSIONS AND FURTHER WORK	207
8.1	Key findings and contributions	208
8.1.1	LIDAR calibration	208
8.1.2	Data collection	210
8.1.3	Segmentation and segment analysis	212
8.1.4	Comparison of classifiers based on supervised learning	217
8.1.5	Evaluation of unsupervised classifier	219
8.2	Recommendations for further work	221
A	APPENDIX: VIDEO DETAILS AND CODE LISTINGS	226
A.1	Video result description	226
A.2	Objective function of calibration chapter	229
A.3	Fusion segmentation	230
A.4	Plane orientation	232
	REFERENCES	233

LIST OF FIGURES

Figure 1.1	Example of semantic labelling	5
Figure 1.2	Stereo vision diagram	7
Figure 1.3	Basic principle of LIDAR distance measurement	9
Figure 1.4	The ibeo LUX LIDAR	10
Figure 1.5	Example of classification results	13
Figure 1.6	Data flow-chart and function contents page	16
Figure 2.1	Binocular stereo vision geometry	29
Figure 2.2	Artificial neural network diagram	33
Figure 2.3	Semantic labelling example from Posner et al. [148]	44
Figure 2.4	Semantic labelling example from Ess et al. [42]	45
Figure 3.1	Sensor fusion configuration	51
Figure 3.2	Results of preliminary Nelder-Mead calibration work	58
Figure 3.3	Target used for LIDAR calibration	61
Figure 3.4	Physical setup of the fusion system	62
Figure 3.5	Captured data from the camera and the laser scanner	63
Figure 3.6	Required target size at a given distance	64
Figure 3.7	Expected vertical position in the camera image	66
Figure 3.8	ibeo LUX scan layer diagram	67
Figure 3.9	Results of calibration with lens distortion model	70
Figure 3.10	Demonstration of calibrated sensor fusion data	71
Figure 3.11	Sensor mounting positions on data collection vehicle	76
Figure 3.12	LIDAR interpolation and range bound estimation	77
Figure 3.13	Improved stereo after integrating LIDAR depth information	78

Figure 3.14	Evaluation of LIDAR to stereo fusion accuracy	81
Figure 3.15	Histogram of error magnitude from Figure 3.14 .	82
Figure 4.1	Land Rover used for data collection	84
Figure 4.2	ibeo LUX four layer LIDAR	86
Figure 4.3	Videre STOC stereo on chip system	87
Figure 4.4	Point Grey Research Bumblebee2	88
Figure 4.5	HP EliteBook 8540w	90
Figure 4.6	Example of image tearing due to insufficient processing power while using H264 compression codec	93
Figure 4.7	Example of decoder noise, contrast greatly enhanced	94
Figure 4.8	Map of route taken for dataset “A46”	95
Figure 4.9	Map of route taken for dataset “RX Residential”	96
Figure 4.10	Map of route taken for dataset “Berkswell Auto”	97
Figure 4.11	Map of route taken for dataset “Berkswell Manual”	98
Figure 4.12	Sample of set “A46”	99
Figure 4.13	Sample of set “RX Residential”	99
Figure 4.14	Sample of set “Berkswell automatic”	100
Figure 4.15	Sample of set “Berkswell manual”	100
Figure 5.1	Example of ineffective contrast enhancement	103
Figure 5.2	Example of grey-world colour correction	104
Figure 5.3	Example of EDISON mean-shift segmentation	106
Figure 5.4	Demonstration of enhanced Canny segmentation	107
Figure 5.5	Illustration of Canny threshold levels	109
Figure 5.6	Secondary edge search for region closing	110
Figure 5.7	Example of binary morphology operations on test image	110
Figure 5.8	Example of closed region Canny edge detection	111
Figure 5.9	Example depth map to demonstrate depth segmentation	111
Figure 5.10	Removal of noise from depth segmentation	113

Figure 5.11	Merging of depth segmentation with edge segmentation	114
Figure 5.12	Outputs from the three segmentation fusion steps	116
Figure 5.13	Merging of small regions from edge segmentation	118
Figure 5.14	Output of segmentation fusion on sample 1	118
Figure 5.15	Output of segmentation fusion on sample 2	119
Figure 5.16	Evaluation of segmentation performance	120
Figure 5.17	Example of RGB colour space	124
Figure 5.18	Example of HSV colour space	127
Figure 5.19	Sample image to demonstrate texture descriptors	128
Figure 5.20	Colour coded magnitude maps showing hue mean and hue STD	129
Figure 5.21	Colour coded magnitude maps showing saturation mean and saturation STD	129
Figure 5.22	Colour coded magnitude maps showing intensity mean and intensity STD	130
Figure 5.23	Colour coded magnitude maps showing intensity range and maximum intensity	131
Figure 5.24	Colour coded magnitude maps showing entropy and vertical position	132
Figure 5.25	Colour coded magnitude maps showing solidity and area of segments	133
Figure 5.26	Colour coded magnitude maps showing horizontal GLCM contrast and correlation of segment	135
Figure 5.27	Colour coded magnitude maps showing horizontal GLCM homogeneity and energy	135
Figure 5.28	GLCM matrix analysis illustration	137
Figure 5.29	Colour coded magnitude maps showing orientation and curvature	138

Figure 5.30	Image and disparity map of road segment used to extract point cloud	139
Figure 5.31	Point cloud extracted from road segment	139
Figure 5.32	Linear surface fitted to road point cloud	142
Figure 5.33	Point cloud rotation and offset prior to polynomial surface fitting	144
Figure 5.34	Polynomial surface fitted to road point cloud	147
Figure 5.35	Colour coded magnitude maps showing height and plane fit MSE	147
Figure 5.36	Example of robust plane fitting, MSE is less than 0.001	149
Figure 5.37	Example of non-robust plane fitting, MSE is 0.03	150
Figure 5.38	Box plot of segment measurement data	152
Figure 5.39	Variance explained by each principal component	154
Figure 5.40	1 st and 2 nd principal components	155
Figure 5.41	1 st and 3 rd principal components	156
Figure 6.1	Software used for manual image tagging	162
Figure 6.2	Deriving target class from pixel tagging	164
Figure 6.3	Class membership counts	166
Figure 6.4	Grid search for SVM parameters	171
Figure 6.5	Determining optimal number of hidden nodes	175
Figure 6.6	Confusion matrix for SVM classifier	179
Figure 6.7	Confusion matrix for neural network classifier	180
Figure 6.8	ROC curve for SVM classifier	181
Figure 6.9	ROC curve for neural network classifier	182
Figure 6.10	Confusion matrix for SVM classifier without “None” class	184
Figure 6.11	ROC curve for SVM classifier without “None” class	185
Figure 6.12	Classifier output example one	188
Figure 6.13	Classifier output example two	189

Figure 6.14	Classifier output example three	190
Figure 7.1	Example of sub-class to super-class mapping	193
Figure 7.2	Accuracy of unsupervised method against sub-class count	197
Figure 7.3	BRR of unsupervised method against sub-class count	198
Figure 7.4	Distribution of super-class membership against sub-class count	199
Figure 7.5	Confusion matrix for unsupervised classifier	201
Figure 7.6	Classifier output example four	204
Figure 7.7	Classifier output example five	205

LISTINGS

Listing 3.1	Optimisation report associated with the results in Figure 3.9	69
Listing A.1	Objective function of calibration optimisation, ϕ	229
Listing A.2	Segmentation fusion	230
Listing A.3	Plane orientation	232

LIST OF TABLES

Table 3.2	Initial values for calibration parameters	68
Table 3.3	Independent samples and resulting projection for calibration validation	73
Table 3.4	Final calibration parameters used during the acquisition of all datasets	75
Table 4.1	Video compression settings for data recording	94
Table 4.2	Summary of "A46" Dataset	95
Table 4.3	Summary of "RX Residential" Dataset	96
Table 4.4	Summary of "Berkswell Auto" Dataset	97
Table 4.5	Summary of "Berkswell Manual" Dataset	98
Table 5.1	Parameters used in EDISON package	115
Table 5.2	Summary of segmentation performance test	121
Table 6.1	Summary of classes and weightings	167
Table 6.2	Summary of classification results	186
Table 7.1	Summary of unsupervised classification results	203

*If we knew what we were doing,
it wouldn't be called research,
would it?*

— Albert Einstein [76]

ACKNOWLEDGEMENTS

It would not have been possible to write this doctoral thesis without the help and support of the numerous kind people around me both within, and outside of, the University of Warwick. It would be impossible to mention everyone who has been involved here, but the following people have been particularly influential and supportive over the 4 years that I have been conducting this research.

Above all I would like to thank my partner Rui Xin and my parents for their unyielding efforts to keep me motivated and working despite my huge reluctance at times! Also my sister, Sally, for putting in all those hours of hand labelling of images for research purposes!

When it came to writing this thesis it would not have been possible without the constant enthusiasm and encouragement from my second supervisor Dr. Emma Rushforth and my initial secondary supervisor Prof. Ken Young who unfortunately could not see my PhD through to its conclusion.

On the academic side, I would like to thank my first supervisor Dr. Yingping Huang for his ideas and great experience in the field of image processing and all of his input during the early stages of the project. On this note I would also like to thank Dr. Thomas Popham and Anna Gaszczak whose contributions to the texture analysis work were invaluable.

Finally I would like to gratefully acknowledge Alain Dunoyer and Paul Nichols from the Jaguar Land Rover group for their support throughout the research phases of this work by allowing access to their equipment.

DECLARATION

I confirm that the work presented in this thesis is entirely my own work, conducted under the supervision of Dr. Emma Rushforth. No part of the work contained in this thesis has been submitted for a degree at any other university.

Coventry, August 2013

A solid black rectangular box used to redact the signature of the author.

Thomas J. Osgood

PUBLICATIONS

Some ideas and figures have appeared previously in the following publications:

- Thomas J. Osgood, Yingping Huang, and K. Young. Minimisation of alignment error between a camera and a laser range finder using nelder-mead simplex direct search. In *2010 IEEE Intelligent Vehicles Symposium (IV)*, pages 779–786. IEEE, June 2010. ISBN 978-1-4244-7866-8. doi: 10.1109/IVS.2010.5548126
- Thomas J. Osgood and Yingping Huang. Sensor fusion calibration for driver assistance systems. In *2011 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pages 187–192. IEEE, July 2011. ISBN 978-1-4577-0576-2. doi: 10.1109/ICVES.2011.5983812
- Thomas J. Osgood and Yingping Huang. Calibration of laser scanner and camera fusion system for intelligent vehicles using Nelder–Mead optimization. *Measurement Science and Technology*, 24(3):035101, March 2013. ISSN 0957-0233, 1361-6501. doi: 10.1088/0957-0233/24/3/035101. URL <http://iopscience.iop.org/0957-0233/24/3/035101>

ABSTRACT

At the highest level the aim of this thesis is to review and develop reliable and efficient algorithms for classifying road scenery primarily using vision based technology mounted on vehicles. The purpose of this technology is to enhance vehicle safety systems in order to prevent accidents which cause injuries to drivers and pedestrians.

This thesis uses LIDAR–stereo sensor fusion to analyse the scene in the path of the vehicle and apply semantic labels to the different content types within the images. It details every step of the process from raw sensor data to automatically labelled images.

At each stage of the process currently used methods are investigated and evaluated. In cases where existing methods do not produce satisfactory results improved methods have been suggested. In particular, this thesis presents a novel, automated, method for aligning LIDAR data to the stereo camera frame without the need for specialised alignment grids.

For image segmentation a hybrid approach is presented, combining the strengths of both edge detection and mean-shift segmentation. For texture analysis the presented method uses GLCM metrics which allows texture information to be captured and summarised using only four feature descriptors compared to the 100's produced by SURF descriptors. In addition to texture descriptors, the 3D information provided by the stereo system is also exploited. The segmented point cloud is used to determine orientation and curvature using polynomial surface fitting, a technique not yet applied to this application.

Regarding classification methods a comprehensive study was carried out comparing the performance of the SVM and neural network algorithms for this particular application. The outcome shows that for this particular set of learning features the SVM classifiers offer slightly better performance in the context of image and depth based classification which was not made clear in existing literature.

Finally a novel method of making unsupervised classifications is presented. Segments are automatically grouped into sub-classes which can then be mapped to more expressive super-classes as needed. Although the method in its current state does not yet match the performance of supervised methods it does produce usable classification results without the need for any training data. In addition, the method can be used to automatically sub-class classes with significant inter-class variation into more specialised groups prior to being used as training targets in a supervised method.

ACRONYMS

ADAS	Advanced Driver Assistance Systems
AGC	Automatic Gain Control
ANN	Artificial Neural Network
BER	Balanced Error Rate
BRR	Balanced Recall Rate
CCD	Charge-Coupled Device
CMOS	Complementary Metal-Oxide-Semiconductor
EDISON	Edge Detection and Image Segmentation
FOV	Field Of View
FPGA	Field-Programmable Gate Array
GLCM	Gray-Level Co-occurrence Matrix
HSV	Hue, Saturation, Value
JLR	Jaguar Land Rover
LIDAR	Light Detection And Ranging
MSE	Mean Squared Error
PCA	Principal Component Analysis
PnP	Perspective-n-Point
RBF	Radial Basis Function
RGB	Red, Green, Blue
ROC	Receiver Operating Characteristic
SRI	Spherical Range Image
STOC	Stereo On Chip
SVD	Singular Value Decomposition
SVM	Support Vector Machine

1

INTRODUCTION

CONTENTS

1.1	Advanced driver assistance systems (ADAS)	1
1.2	Basic concepts	5
1.2.1	Semantic labelling	5
1.2.2	Stereo vision	6
1.2.3	LIDAR	8
1.2.4	Supervised learning	11
1.2.5	Unsupervised learning	12
1.3	Aims and contributions of thesis	12
1.4	Thesis outline	15

1.1 ADVANCED DRIVER ASSISTANCE SYSTEMS (ADAS)

Cars have become safer for their occupants in recent years due to passive safety features installed by most of the major motor manufacturers in the EEC. Because of this the chances of being seriously injured in a crash have diminished, however, with the recent surge in the number of road users the frequency of collisions continues to rise. More cars on the road means less room for driver error so, consequently, despite passive safety systems there has been little change in the number of collisions leading to personal injury. In fact recent road safety reports from the Department for Transport [53, 178] show a small increase in driver casualties since 2010 in the UK.

Passive systems such as air bags and crumple zones provide a reasonable level of protection to the driver and passengers after impact but, the only sure way of preventing road user injury (and damage to property) is to prevent collisions from occurring in the first place. While some accidents, such as mechanical failure, are truly beyond the driver's control, the vast majority could be avoided if road vehicles had the ability to predict collisions and take evasive action. This is the principal aim of Advanced Driver Assistance Systems (ADAS) - systems which assist the driver by detecting and correcting driver mistakes. Some examples of ADAS systems that already exist are lane departure warning systems and automatic parallel parking systems.

The idea for an ADAS which predicts and prevents collisions is not new, several research groups are already building such systems at the time of writing (e.g. PreVent [180]). All ADAS systems involve the interaction of many sub-systems from different engineering disciplines including software, hardware and mechanical aspects, however the two most important components are:

1. The sensing of the area surrounding the vehicle: A vehicle that can predict collisions would need to have information about what is happening in its immediate surroundings, particularly regarding other vehicles, pedestrians, solid obstacles and even road signs. These data can be collected in various forms via sensors that already exist, or will soon exist on commercially available vehicles. The purpose of collecting and analysing the data is so that other projects, such as PreVent, can integrate them into systems which perform the second important step in the collision avoidance ADAS below.
2. Using sensor data, and perception of the environment, to decide if the current situation is safe: If the situation becomes unsafe – for example the car ahead has braked suddenly and the driver has not started braking in time to avoid a collision, then the ADAS can step in and apply the brakes automatically. This step involves

physical intervention in the control of the vehicle including computer controlled brakes and steering override. This aspect of the system will not be covered in this thesis.

Going back to point 1, there are a number of ways of collecting this information, using several types of currently available sensor. When considering which sensors are suitable for this application the priority is to make a selection that can be easily and economically fitted to the majority of road vehicles. The performance of the sensors is the secondary consideration, at the moment, since it will be some time until 3D rotating Light Detection And Ranging (LIDAR) devices, for example, can be fitted to every vehicle at a low cost (presently even single plane LIDAR devices such as the LUX are in the region of £14,000 [64]). However, history has shown us that new technologies quickly become more affordable.

One good choice is to use video cameras – they are so cheap to produce now that some production vehicles, such as the new Land Rover line from Jaguar Land Rover (JLR), already come fitted with a set of cameras that capture a 360° view of the surroundings. A single camera however can only provide the driver with the means to see behind the vehicle. A pair of cameras, on the other hand, can be used to estimate size, shape and distances using stereo techniques discussed in the next section. Ultra-sound devices are also fitted to the majority of modern cars to assist in parking and manoeuvring however, due to their extremely short range, they are not applicable to collision avoidance at normal driving speeds. The other choices for active distance sensing are radar and LIDAR. Since LIDAR outperforms radar in terms of accuracy and resolution it is generally favoured in ADAS research.

So, if the goal is to “sense” the surroundings and manipulate the data into a form that allows an ADAS to make decisions about the current environment, then simply gathering and recording images from cameras is not sufficient. An image, to a computer, represents

nothing but a matrix of values. There is no information about the size, shape, distance and properties of any object seen in that image. On the other hand, the human eye can infer all that information and more just by looking at a single snapshot. The difference lies in the fact that humans can recognise the objects that compose the scene; they know how each of these objects behaves, what size it is, how fast it might be moving and whether it should be a cause for concern, if it appears in the path of the vehicle. In order to help computers determine the more difficult attributes, such as size, sensors such as stereo cameras and [LIDAR](#) devices can be used to reconstruct the scene in 3D allowing real world measurements to be made.

Vehicles capable of sensing their surroundings are not only of interest to car manufactures for safety systems, but the underlying systems are also applicable to autonomous space exploration e.g. the Mars rover [\[115\]](#), military applications e.g. the DARPA challenge [\[171\]](#) and fully autonomous passenger cars e.g. [\[120\]](#). The ability to autonomously detect and avoid pedestrians, for example, would be the next step in the suite of existing vision based driver assistance technologies such as road sign detection (e.g. Broggi et al. [\[22\]](#)) and lane departure warning systems (e.g. McCall and Trivedi [\[117\]](#)).

The main goal of this thesis is to explore all the tasks involved in the processing of raw sensor data into scene description which is meaningful to a computer (i.e. scene parsing). This starts with the selection, configuration and evaluation of current vehicle sensors. Then the processing and identification of the collected data. The project will evaluate a range of currently used techniques in the field of image processing and classification. In areas where information is currently lacking, such as a comparison between classification techniques, further investigation is carried out. Where current techniques do not provide results ideal for this application, improvements have been suggested.

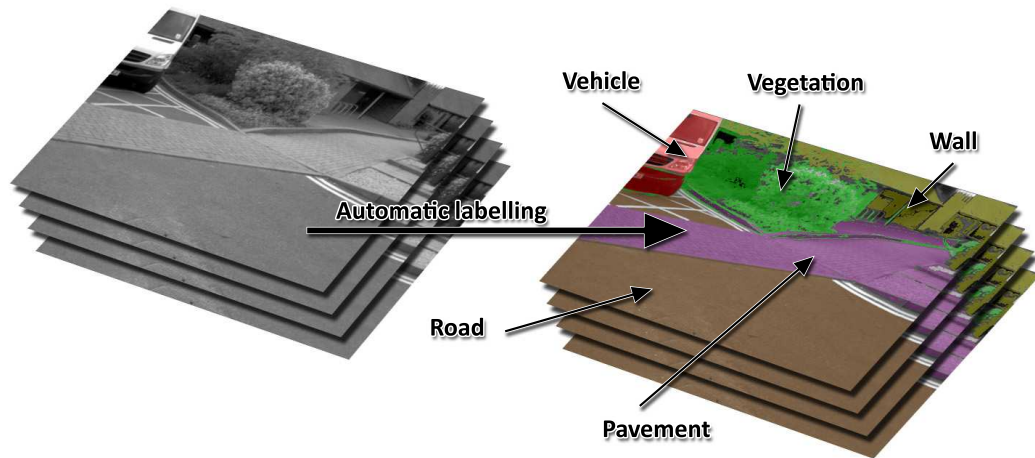


Figure 1.1: Example of semantic labelling.

1.2 BASIC CONCEPTS

1.2.1 *Semantic labelling*

Semantic labelling, also known as scene parsing or full screen labelling, is the process of identifying the content in every region of an image. The idea is partly inspired by the way a human perceives a road scene – the goal being to mimic this understanding and to separate the raw data into individual surface types that make up the surroundings. If this can be done, then the data from the sensors can become more than just data, they will become a description, or labelled map of the environment, which can easily be queried to decide, for example, if the space ahead is safe to drive in. So, in the same way that a human can look at an image and identify pedestrians, cars, trees, pavements etc. in the image, this data analysis system will attempt to identify the same regions of the image and label them as that object.

A simplified step-by-step description of this process would look like this:

- A. Sensors on the vehicle collect raw data in the form of video (image) data. These images will depict the view in front of the vehicle (for example).

- b. Images will be processed to find edges and boundaries, this is known as segmentation. Simplistically this would be done by detecting lines in the image that separate regions of different colour, or brightness.
- c. For each of the segments identified in the previous step, take measurements and make observations on the content of the segment which might help identify what this region contains. This might include measurements such as the colour, the brightness and the texture of the image segment.
- d. Supposing that one has some corpus of data which describes the relationship between these observations and what type of object they relate to, then: given all the information that has now been collected about each segment, the content can be identified, or matched, to a known type of object. As a simplified example, suppose it is known that segments that are mostly grey contain road, then any segments matching this description can be labelled as road. The more information that is available the more precise the prediction will be.
- e. The result will be a labelled version of the original image and all the associated information these labels infer. See [Figure 1.1](#) as an example.

This process will be referred to as semantic labelling.

1.2.2 Stereo vision

Stereo vision is obtained using two cameras which are displaced horizontally from each other in order to obtain different views of the same scene. Depth information can be obtained by examining the relative positions of objects in the two perspectives. Objects which are closer to the cameras will have a greater difference (disparity) in apparent position between the two perspectives. A disparity map can be formed from the combination

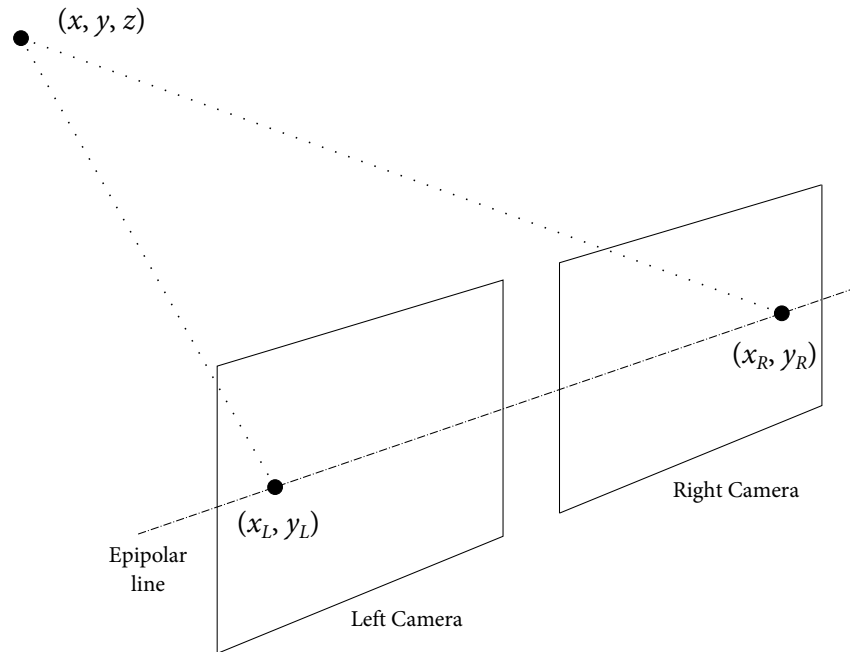


Figure 1.2: Stereo vision diagram – (x, y, z) is a point in real space, (x_L, y_L) is a point on the 2D projection image captured by the left camera, (x_R, y_R) is a point on the 2D projection image captured by the right camera.

of the 2 images that will allow a system to make decisions based on the distance of objects from the vehicle.

This is seen more clearly in [Figure 1.2](#) which shows the same point in 3D space being projected onto the left and right camera images. If you imagine the point is very close to the cameras then the horizontal position in the left and right images will differ greatly whereas if the point was on the horizon it would appear in practically the same spot on both images. This also illustrates the main drawback of stereo camera systems - the disparity resolution is inversely proportional to the square of the distance from the camera. The actual performance is dependant on a number of factors such as the resolution of the sensors, the Field Of View ([FOV](#)) and the distance between the cameras.

1.2.3 LIDAR

The LIDAR sensor is used to accurately measure the distance to a given surface using pulses of laser light. Although it has the same purpose as the stereo vision system mentioned above it has several useful advantages in the application of driver assistance systems. Firstly it is an active sensor as opposed to a passive sensor meaning it generates its own light and is therefore not dependent on the ambient lighting allowing it to work effectively at night and to some extent in foggy weather. Secondly it is significantly more accurate and capable of measuring surfaces much further away from the vehicle than is possible with a stereo vision system. As a rough example a good stereo vision system with high resolution cameras would have a mean error of 2.5m at a distance of 30m. By contrast a LIDAR device can scan a surface at 200m away with an uncertainty of less than 10cm [64].

The main reason that LIDAR devices are not deployed commercially in vehicles is the prohibitive cost. At the time of writing, a LIDAR with suitable performance would cost in the region of £14,000 meaning it is simply not economically viable to fit them to personal vehicles. There are, however, new developments which show that the complex spinning mirror mechanism can be replaced with solid state components allowing the price to be reduced to only a few hundred pounds which would make the prospect of widespread use much more likely. It is for this reason that they have been included in this research.

In its most basic configuration a LIDAR is nothing more than a laser range finder. This device has a laser emitter and a detector, a small laser pulse is fired from the emitter, reflected by a distant surface then collected at the detector. The total time for this journey is recorded with high precision (since it will generally be in the order of nanoseconds) then, using the known constant speed of light, the distance travelled is calculated and half this distance is the range of the target.

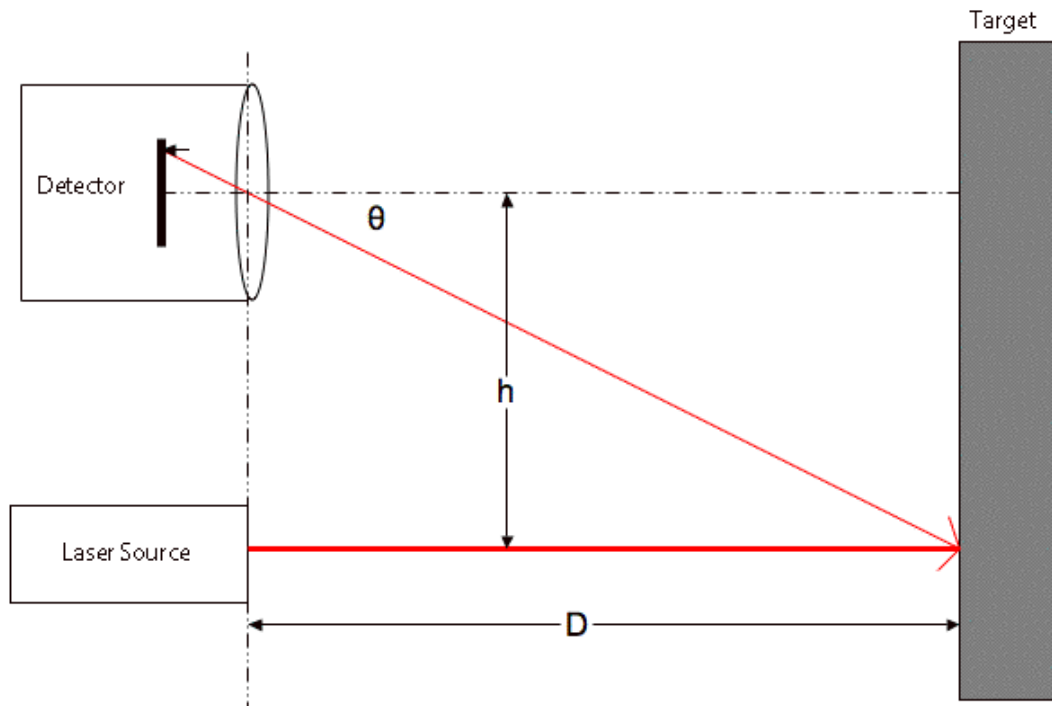


Figure 1.3: Basic principle of LIDAR distance measurement. Simple figure to show fundamental operation behind a laser range finding device, θ , h , D are the angle, separation between source and sensor and the distance to the target respectively. D is calculated by application of trigonometry. Source: http://www.codeproject.com/KB/cs/range_finder/laser-range-finder-1.gif.

Another implementation of a laser range finder is to project a laser dot on the target then use a conventional CMOS camera to identify the illuminated point. Using the known point of reflection and the relative positions of the laser emitter and camera the distance can be calculated using trigonometry as shown in Figure 1.3.

To extend this theory to the more sophisticated LIDAR models on the market today, spinning mirrors are used to sweep the laser beam from side to side. Using 2 mirrors the laser can be scanned horizontally, and vertically, allowing entire rectangular regions of a scene to be scanned. Using precise timing, and knowledge of the mirror's motion, the path of each laser pulse is known, allowing the computation of the exact 3D point where the laser was reflected. Since all the laser pulses are reflected back into the detector simultaneously methods must be employed to uniquely identify each pulse so as to know

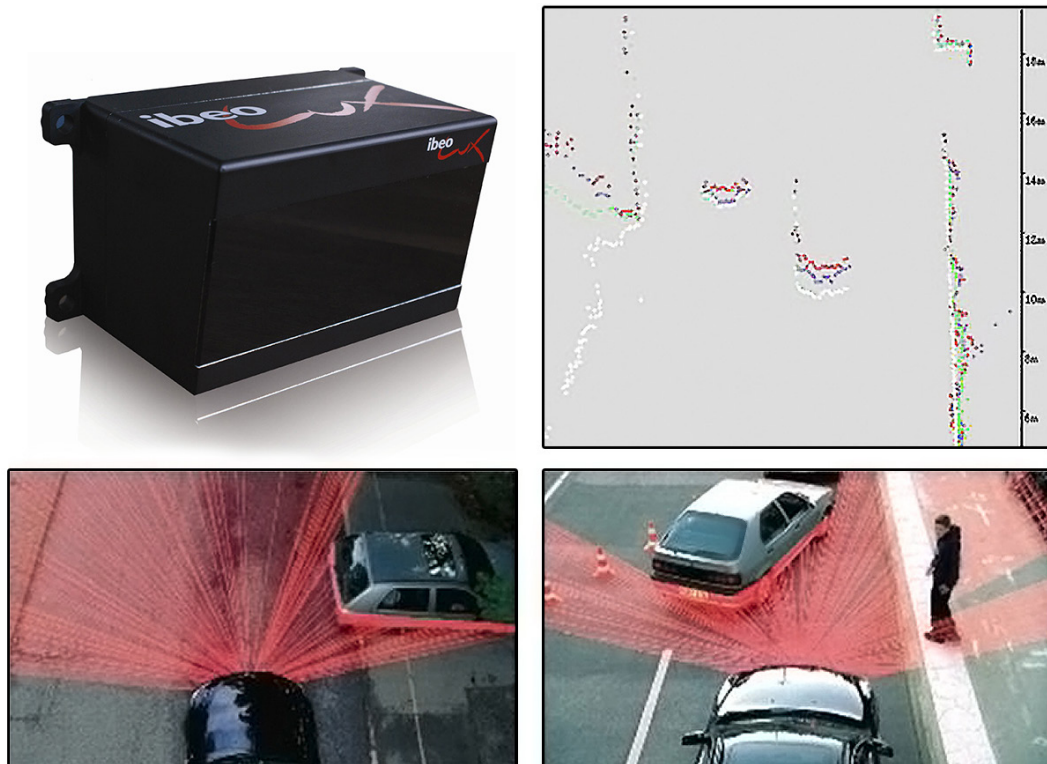


Figure 1.4: This figure illustrates the operation of a scanning [LIDAR](#), in this case an ibeo LUX device. Laser pulses are emitted in a scanning arc pattern and are received by a detector simultaneously allowing many scans to be performed each second producing a distance graph shown in the top right [\[64\]](#). Pictured are various marketing images from the IBEO website, the top-right LIDAR scan does not correspond to the illustrations.

the particular orientation the pulse originated from. This is generally done using a unique pattern of frequency modulation on the laser carrier frequency for each pulse however the exact implementation is proprietary [\[60\]](#).

[Figure 1.4](#) gives an idea of the performance, and output, produced by commercially available units, the ibeo LUX can scan at 50Hz at ranges up to 200m with 0.25° angular resolution over a 110° arc [\[64\]](#). It does this over 4 horizontal layers simultaneously. The figure shows the scan map in the top right where the colours indicate the scan layer.

1.2.4 *Supervised learning*

Supervised learning is a type of machine learning which, given a set of input data and a matching set of output data, attempts to reproduce the mapping from input to output. The hope is that the learning algorithm can infer how the given outputs are determined by, or influenced by, the given inputs. Then, using the same inference, determine the correct output for future inputs which were not present in the training data. The input data generally take the form of a description vector, that is a list of numerical attributes describing a sample. For example, in speech recognition, a sample might be a single spoken word and the descriptors could be pitch, duration, amplitude, number of volume peaks etc. The output is generally a single value, such as a class ID indicating which category the sample belongs to, or a floating point number.

As hinted at above there are two main types of supervised learning: classification and function approximation. Function approximation might be used to predict the result of a chemical reaction given quantities of reactants, ambient temperature, presence of catalysts etc. allowing the results of any combination of these factors to be estimated without testing each combination exhaustively. Neural networks are particularly good at approximating general functions of this type where the output can be any real number.

The other type is of more relevance to this work. The classifier attempts to determine the correct class of a sample given previous examples of correct categorisation. The simplest implementation is a binary classifier which will determine if each sample is in a set or not in the set. For example a classifier trained to detect the presence of a fault in manufactured goods would give the output of yes or no, nothing else. There are also multi-class classifiers which attempt to assign each sample to the best fitting class. For example within the scope of image classification (discussed earlier and of relevance to this research), given properties of a small segment of an image the algorithm attempts to determine the correct classification such as road, grass, sky etc.

1.2.5 *Unsupervised learning*

Unlike supervised learning, unsupervised learning does not require any training data in order to group observations into separate classes. For this reason it is better described as data clustering. It may, however, have a learning element which allows cluster definitions to be continuously updated to include information gained from new samples passed through the system.

For example, an unsupervised learning algorithm may be used to group measurements from hundreds of leaves from three different plants into three groups without any other prior knowledge. It can do this by looking for similarities in the given samples, then dividing them into three groups such that members of the same group are as similar as possible, while being as dissimilar as possible to samples in the other two groups.

If these groups were determined using an initial bulk of data, then the resulting group definitions were used to classify further samples introduced at a later time, then these samples could be used to simultaneously update the group definitions as classifications are made.

1.3 AIMS AND CONTRIBUTIONS OF THESIS

This research aims to produce a working prototype of a system which can take raw vehicle sensor data and use it to produce a fully labelled description of the scene ahead. If this information can be reliably created then it would make possible several major advances in driver assistance and safety systems. [Figure 1.5](#) shows an example of the results produced by this prototype. Please refer to [Figure 1.6](#) for clarification on the steps involved and how these elements fit together.

Because the primary goal is to have this applied to personal vehicles the possible solutions must be restricted to those which can be fitted economically and do not need significant

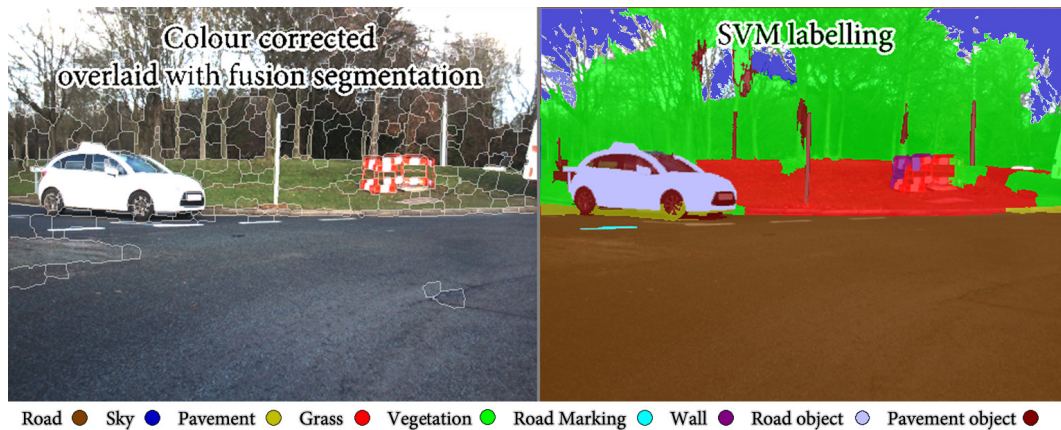


Figure 1.5: Example of classification results which will be presented in [Chapter 6](#). The image on the left has been automatically segmented using the fusion segmentation method presented in this thesis (white lines show segmentation boundaries). Each segment has then been analysed to determine texture, colour and 3D attributes. These are then fed into a trained SVM to produce the segment classification seen on the right. The types of surface which can be predicted are shown along the bottom.

configuration from the user. Based on prices today this would exclude the LUX [LIDAR](#) from being used in this research however representatives from ibeo, the manufacturers of the LUX, seem confident that it will not be long until solid state [LIDAR](#) devices can be produced. The solid state models would be able to deliver similar performance to current devices but at only a fraction of the cost. For that reason this research assumes that affordable [LIDAR](#) devices will soon be common so the benefit of using them should not be excluded from current research.

Also, because it would be ideal for the system to function without requiring significant amounts of human training, an unsupervised approach is also investigated to see if possibilities exist in this direction. With these aims in mind the following contributions have been produced during the course of this research:

- A novel method of calibrating a [LIDAR](#) and stereo camera for the purpose of sensor fusion. The published method allows an unrectified camera system to be aligned with a [LIDAR](#) projection without the need for any particular test pattern or precise measurements of the mounting points. The method will report the necessary

reverse distortion model for the camera and the necessary transformation matrix to project [LIDAR](#) measurements directly onto the rectified or unrectified image stream (see [Chapter 3](#)).

- A novel hybrid image segmentation method was developed, it combines a Canny edge detector (modified to create closed regions), the Edge Detection and Image Segmentation ([EDISON](#)) [[32](#)] mean-shift segmentation algorithm and a depth map segmentation. This method combines the main strengths of the mean-shift algorithm; the clustering of areas of similar texture and of the Canny edge detector; preserving weaker and softer edges between regions. Together these segmentations are fused with the depth map segmentation which separates physical surfaces and creates regions which do not span multiple types of content (see [Section 5.2](#) on page [105](#)). [Figure 1.5](#) shows an example of an image segmented using this method.
- Demonstrate the use of combined visual and depth derived segment features to describe image segments. The novel use of polynomial surface fitting on extracted point cloud of each segment allows for differentiating between flat man-made surfaces and more natural curved or sweeping objects. Gray-Level Co-occurrence Matrix ([GLCM](#)) texture descriptors greatly reduce the dimensional size of the training data compared to similar techniques giving a solution which is capable of classifying a scene in real time (see [Section 5.4](#) and [Section 5.5](#)).
- A comprehensive comparison study was performed to explore the difference in performance between the two most commonly applied supervised classification methods, in the context of road imagery classification. No existing study comparing Support Vector Machine ([SVM](#))s and neural networks, on a dataset comprised of texture and spatial training features, could be found. Manually labelled stereo images were used to train a set of [SVM](#) classifiers and a neural network using best available practices. The two systems were trained on a mixture of data from two

different driving environments; urban and rural locations, then the results are compared (see [Chapter 6](#)). [Figure 1.5](#) shows an example of classification results obtained from the [SVM](#).

- A novel classification method capable of producing acceptable results using unsupervised learning which is an area under-explored in current literature. Segment descriptors are clustered into a large number of sub-classes then the only human interaction needed is to identify which sub-classes should form the members of a true class (super-class). Hand tagged images are used to provide performance data on the technique (see [Chapter 7](#)).
- In addition to the primary contributions above, this research has made available a collection of matched stereo and [LIDAR](#) data sets along with several batches of hand tagged images for use in future projects and performance comparisons. Also to help newcomers the data collection chapter documents, the common practical difficulties encountered when setting up a data collection platform and provided solutions for future researchers looking to continue this work (see [Chapter 4](#)). Due to the large size of this data it is currently available on request only.

1.4 THESIS OUTLINE

This section will briefly describe the content and motivation of each chapter. For a visual, one page summary of the various sub-projects involved in this thesis, and how they fit together, refer to the flow chart in [Figure 1.6](#) on the next page.

CHAPTER 1 This introduction gives an outline of the main motivations of this research and gives a brief description on some of the core aspects. It also lays out the primary contributions of the thesis and provides a guide to the contents of each chapter.

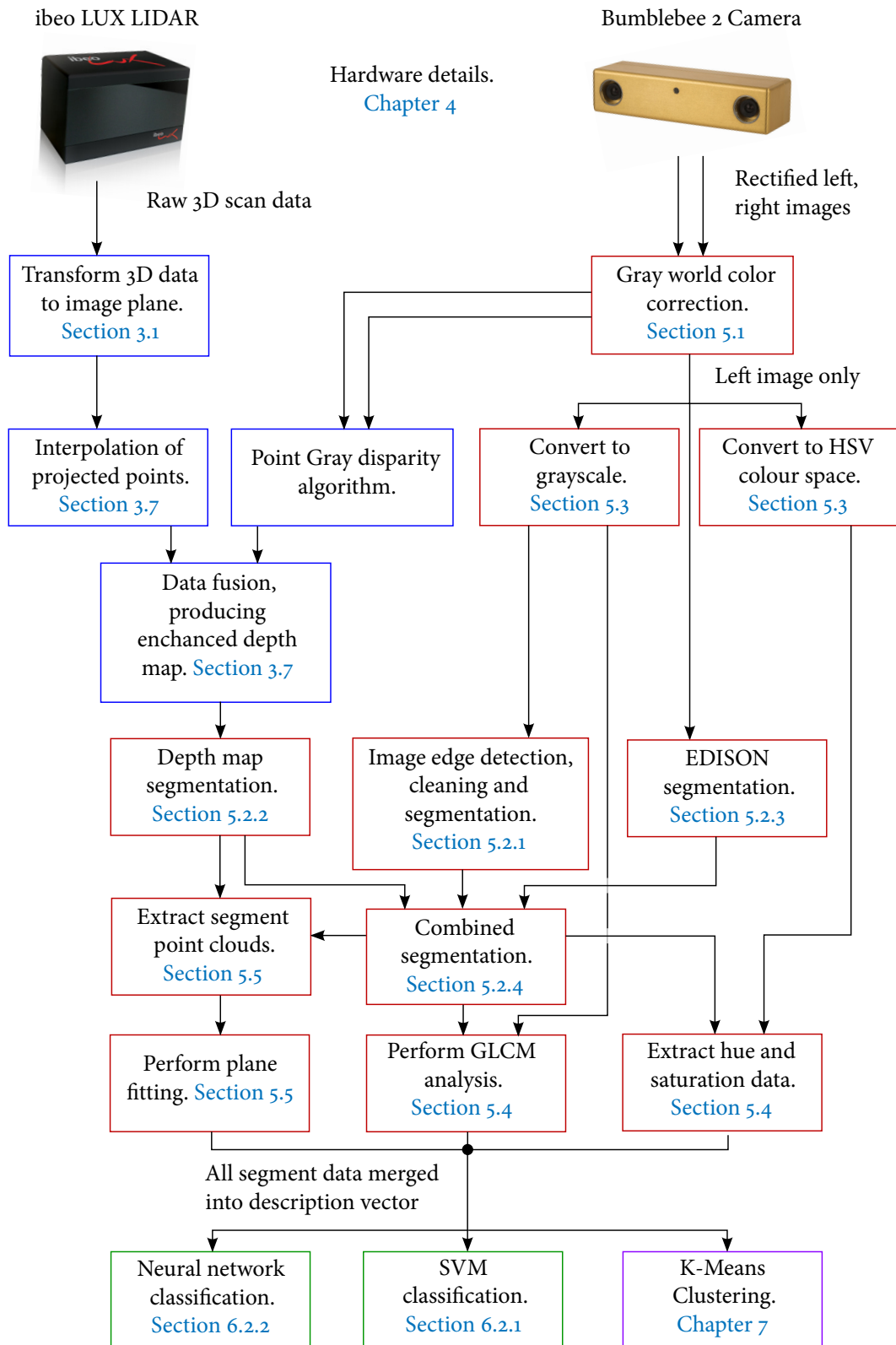


Figure 1.6: This flowchart maps the most significant steps in the data processing from raw inputs to segment description vectors. Section links are provided for each step for quick navigation. Blue boxes relate to **LIDAR** data collection and alignment. Red boxes relate to image processing, segmentation and analysis. Green boxes relate to using the data for classification and the purple box relates to the novel unsupervised method.

CHAPTER 2 The literature review gives specific details on the work of other authors which has formed some part of the foundation for this research. This includes methods which have been incorporated into this work in either modified or unmodified forms and the conclusions of others which have influenced implementation decisions during this work. Of particular importance is the theory of depth from stereo, sensor calibration for sensor fusion, machine learning and previous works on semantic labelling.

CHAPTER 3 This chapter presents a novel method for determining both intrinsic and extrinsic calibration parameters required for accurately projecting **LIDAR** scans onto the image captured by the camera. The method described here was also published in *Measurement Science and Technology* [139]. This necessary step allows the stereo camera and the **LIDAR** to be used together to produce a more accurate depth map of the scene. The sensor fusion technique that merges the two data sources is presented along with examples and a short evaluation of the accuracy.

CHAPTER 4 This chapter on data collection first describes the hardware used for the experiments including some devices which were evaluated and found to be unsuitable for this application. Secondly some of the challenges encountered during implementation are discussed and how they were overcome. Finally a brief description of each of the datasets created is given and some samples to give an idea of their contents. The datasets cover the three primary road environments, motorway, rural and urban settings.

CHAPTER 5 This chapter details how images are pre-processed, segmented and analysed. There are several steps to the segmentation using both image and depth data. A hybrid implementation is presented using the strengths of each segmentation method to get the most effective result. Each type of segment feature, which will be later used for learning, is also described including how it is calculated. Finally

an investigation into the information provided by each metric is presented using a Principal Component Analysis ([PCA](#)).

CHAPTER 6 This chapter presents the first experiment which focuses on using supervised learning for segment classification. It details how data is hand tagged to provide learning targets, how training and testing sets were chosen and provides details of data manipulation. It then gives a comparison of different learning methods including [SVM](#) and neural networks before evaluating the relative performance.

CHAPTER 7 This chapter presents the second experiment using unsupervised learning to attempt to automatically cluster the segments into similar groups. Details on clustering methods are given followed by a performance evaluation using the hand tagged data.

CHAPTER 8 The final chapter will sum up the main conclusions and findings in each of the preceding chapters. There will also be recommendations for future work to address any limitations that were encountered during this research.

APPENDIX A This appendix contains some potentially useful snippets of Matlab code which can be useful to others as stand alone functions without heavy dependence on external data, other custom functions or environment configuration. It also has a description of the included result video, giving some details on how it is made and the purpose of each section. The video can be found on the included CD or online at the address given in [Section 6.3](#) on page 177.

2

LITERATURE REVIEW

CONTENTS

2.1	Depth from stereo	19
2.1.1	Image capture, preparation and pre-processing	20
2.1.2	Feature matching	23
2.1.3	Verification of matching	27
2.1.4	Distance calculation	28
2.2	LIDAR to camera calibration	30
2.3	Machine learning	31
2.3.1	Artificial neural networks	31
2.3.2	Support vector machines	34
2.3.3	K-means clustering for unsupervised learning	36
2.4	Image processing	38
2.4.1	Edge detection	38
2.4.2	Segmentation	40
2.5	General semantic classification	41

2.1 DEPTH FROM STEREO

In the field of mobile robotics and autonomous systems there is an ever growing number of applications where designers need their products to interact with the real world in a robust and adaptable fashion. Unfortunately the real world, as opposed to controlled

environments, is unstructured and littered with obstacles. In the case of Advanced Driver Assistance Systems (ADAS) it is a common desire of vehicle manufacturers to have a system that can monitor the road ahead and alert the driver to any potential hazards, which is both reliable and economical. It is only recently that integrated computing systems and sensors have reached a point where they perform adequately, are sufficiently affordable, and are robust enough for motor vehicles.

In theory, using only a pair of inexpensive cameras and an integrated computer, such a system would be able to know as much about the road ahead as the driver of the car. In practice however, the problem of extracting meaningful data from cameras alone is far from trivial and consists of a number of processing steps each of which still have ongoing difficulties.

There are a number of approaches to the *depth from stereo* problem, but all of them boil down to the same fundamental objective: find the same point in the left and right images that corresponds to the same physical feature and then measure the linear disparity. With this information it is then a simple case of using triangulation to determine the distance of this point from the cameras, see [Figure 1.2](#) on page 7. The difficulties arise in how these points are identified and verified to correspond to the same object.

The following sections will describe the steps common to all depth from stereo techniques and discuss some of the benefits and limitations of the various existing approaches.

2.1.1 *Image capture, preparation and pre-processing*

In the most basic case images are captured from a pair of cameras which are mounted side by side within a rigid housing (this configuration is used throughout the thesis since pre-built rigs are readily available for purchase). It is desirable to have the camera's shutters synchronised such that each camera captures a frame simultaneously thereby ensuring

that the images captured are at the same instant in time. The horizontal offset between the cameras, as well as the optical characteristics, are required to properly calibrate the system and rectify the captured images. See [Section 3.8](#) on page 79 for full details on this.

The image preparation largely depends on how points will later be matched, but generally it involves identifying regions, or points of interest, in each of the images to be used for depth recovery. The earliest experiments used an area matching approach, so pre-processing was a case of using operators to identify potentially significant portions of the image, examples of this can be seen in papers by Gennery [63] and Moravec [123]. The operation used in the latter calculated the variation in pixel intensity alone in each orthogonal and diagonal direction, through each point in the image using the 8 neighbouring pixel values. The required output of the operator was the smallest of the 4 directional variations. After this operation is applied to the entire image, local maxima are then designated points of interest and the region around each one becomes an image patch. The idea is that the operator will respond to uniquely identifiable features in the image and these features should appear in both images such that a correspondence can be established. The approach does produce reliable matching points, but in most cases the quantity and distribution of the features is far too sparse to be sufficient for any kind of reconstruction, or object identification.

An approach that produces a significantly greater number of features is edge detection. Edge detection algorithms are used to locate discontinuities in image intensity which correspond to the boundaries of physical objects captured in the image. These edge segments can then be matched between left and right images.

Edge detectors return a set of pixels that are determined to be edges and these pixels are then used in some form of matching process, however there are other approaches that aim to match more abstracted geometric features instead of raw pixel data. For example Medioni and Nevatia [118] used the results of an edge detection pass to fit line segment

paths along the strongest edges which can be expressed as a set of vectors instead of a pixel map. In this paper the Nevatia-Babu algorithm [131] was used to find the best fitting set of line segments which can be expressed as pairs of endpoints. Each line segment has an associated pixel contrast comparing the intensity of pixels on either side of the line segment i.e. the difference in pixel values along a line normal to the segment. The information, along with the implied line orientation, was used to find matching line segments in the second image.

A similar approach was used by Ayache et. al. [11, 12, 73] to match line segments to the edge map produced by the Marr-Hildreth [110] and Canny [25] edge operators. Compared to Nevatia-Babu [131] this approach makes improvements by using the two edge detectors to cross verify edge points and filter out weak edges that have a lower probability of appearing in both images. Furthermore the chain of line segments found by this step are not used directly in the matching strategy, instead each chain of line segments is first approximated using a polynomial curve which further reduces the amount of data to be considered for matching. Instead of having features consisting of many pairs of line end points, this approach, instead, gives a set of polynomials described by their coefficients, mid-point and orientation.

The key benefit of the line segment approaches is greatly reduced feature data. Instead of attempting to match up pixel edge maps, which may contain thousands of points, a small group of polynomials representing the strongest edges can be compared in only a fraction of the time. Whilst this was an important consideration in the 1980's when these techniques were developed, the computers of today are orders of magnitude faster and can work directly with the raw image data, in real time. Since working with raw data can potentially provide more fidelity in depth maps, compared to abstracted line segments, this type of simplification is no longer necessary.

2.1.2 *Feature matching*

Once features such as patches, edge points and line segments have been found in each image the next step is to determine if a feature in one image can be matched to a feature in the second image. A match in this case indicates that the feature point in each image lies on the projection of the same physical point in space. If this is the case then the horizontal displacement between the feature across the two images can be used to determine the distance from the observer (see [Figure 1.2](#) on page 7).

Generally, matching strategies depend on the type of features used, camera geometry and configuration. There are many ready built multi-camera rigs available including the most simple binocular setup, trinocular and even camera grid setups composed of over nine cameras. In the cases where more than two cameras are used it is possible to sample a scene from more directions and angles, particularly using a grid of cameras where there are cameras with vertical as well as horizontal offsets. With more samples available, matching strategies are not confined to looking for a single match in a second image, instead there could be a possibility from each image allowing more robust and best fit matches to be made. With this increased information it is possible to perform accurate 3D surface tracking across the entire workspace as shown in the work by Popham [146].

The most common setup is a dual camera (binocular) stereo system with parallel optical axes. Each camera is calibrated such that optical distortions, such as fish-eye effects, are removed and small misalignments in mounting are accounted for. Assuming the calibration is correct then it is known that scan lines in each camera image will cover the same horizontal strip of projected scenery. This means that only a horizontal search is needed making any matching strategy significantly simpler and more efficient.

In this set-up area-based matching is the most basic approach, where the strategy is to find a match for each and every pixel. The area used in this search is a horizontal strip of pixels centred on the pixel being examined. It is necessary to match the entire area in

order to verify a match, since a single pixel value alone could exactly match any number of pixels in the second image i.e it does not provide sufficient information alone. The general approach is to start at the beginning of the first row and select each pixel with its neighbouring column pixels, then superimpose this strip of pixels at each horizontal position on the second image until the minimum difference in intensity or maximum correlation is found. In general the direction of the displacement will be the same for all pixels, but in practice the strategy is to search in both directions simultaneously i.e. alternately taking one step in each direction.

This brute force approach is effective at matching pixels providing that the images are uniformly lit and each camera has identical intensity level quantisation. In practice this is not often the case as cameras have differences in their sensors that will lead to intensity variation, even when capturing the same scene in the same lighting conditions. The most common solution to this problem is to use the difference in neighbouring pixel values instead of the pixel values directly. This is generally done using a Sobel filter on the image before area matching takes place. By eliminating the bias introduced by the individual sensor errors caused by small differences in contrast, illumination and gain control can be mostly ignored.

A more sophisticated method described by Gehrig and Franke [62], which aims to further increase robustness and estimate disparity to sub-pixel accuracy, uses curve fitting on the pixel intensity values. A low order polynomial is fitted to the values to be matched and to each same-sized window in the destination image. The curve segments are then compared to find the best match, the precise offset between the curve's saddle points then gives the disparity to an accuracy greater than a single pixel, since the curves are not restricted to integer number space.

Feature based matching does not match pixel values directly, instead it matches derived points of interest, such as line segments as discussed above. This method is significantly

less prone to false matches and more stable in the presence of illumination differences and camera calibration errors. Features generally consist of a group of points in the image and so the disparity calculation can be done on the centroid of these points, again allowing sub-pixel accuracy. Another benefit of feature based matching is that the analysis of the raw pixel data has already been done by the image pre-processing so the matching of detected features requires negligible processing time, however the total time taken by the more complex pre-processing and simple feature matching may not necessarily be less than the time taken to use area based matching on the entire image. These benefits are unfortunately overshadowed by the somewhat poor disparity density obtained using this approach. i.e. the number of confirmed disparity points in the image will only be as great as the number of features found, which is generally significantly less detailed than the results obtained via area based matching.

The two greatest difficulties encountered in stereo matching, regardless of the methods used, are occlusion and featureless surfaces. The occlusion problem comes from the fact that each camera will see a different patch of scenery behind a close object, the effect can be easily seen by holding some paper near your face and alternately closing each eye whilst observing what can be seen beyond the paper. If the left camera is to be used as the reference image for patch matching, and can see some object that is totally hidden in the view from the right camera, then there is no possible way that the area can be matched between images. This will leave a gap in the disparity map and gives close objects a border of missing measurements. Research on this has been met with some success such as the work by Sun et al. [167].

The second problem occurs when there are regions of the image that are uniform in colour and intensity, such as areas that are under or over exposed, or regions that simply lack any texture such as a clear sky. Any feature detection algorithm will fail to find any kind of uniquely identifiable point in a region of identical pixels and any area matching algorithm will match every point in the region to the exact same position in the other

image, as it will appear to be an identical match. The result is that these areas will appear to have a disparity of zero which translates to a distance of infinity and are generally filtered out by post processing, again giving gaps in the disparity map.

One way of reducing noise and filling in gaps caused by occlusions near discontinuities in scene depth is to apply the assumption that surfaces are mostly smooth i.e. there are not sudden changes in the curvature of the surface. If this assumption is true then it should be reflected in the disparity map as smooth transitions between each disparity level. When there is a large step in the disparity level it can be assumed that one surface has finished and another has begun at a different distance from the camera. If there is a region of missing depth information, between the two disparity levels, then the more distant disparity value can be used to fill in the gap as it is assumed that the distant surface is being occluded by the closer surface. A more sophisticated solution to this problem was proposed by Woodford et al. [187] which aims to produce global stereo. In the case of regions with missing disparity, due to lack of texture, then it is often the case that the edges of the missing region will have similar disparity values since the edges of the texture-less surface can be more easily matched. If this is the case, then the missing region can be filled with a smooth gradient from the disparity at one edge of the region to the other in both directions.

The smoothness constraint can be improved by assuming surfaces conform to a series of piecewise spines instead of a linear surface, by fitting curves to the disparity values available, then the missing values can be found by interpolating using the curve, or a sharp step in the case of an occlusion. The method used will depend on the smoothness threshold, or maximum change in surface curvature allowed. This refinement step is often referred to as global matching with the goal of obtaining a complete depth map of the scene with no omissions.

2.1.3 *Verification of matching*

Due to the repeating patterns present in the structure and appearance of many surfaces, both man made and natural, there are often cases during local matching where there are several equally strong match candidates for a single patch. For example if the left image is the reference image then there could be several patches in the right image that could make a perfect match. This creates an unresolved ambiguity when trying to determine the true disparity of the area in question. Similarly areas that fluctuate wildly between high and low density are unlikely to represent accurate measurements. To address these scenarios, where disparity does not reflect what is possible in a physical scene, several authors have suggested a systematic set of rules and constraints to ensure disparity results are rational [121].

A method of verifying disparity maps, inspired by biology, was developed by Marr and Poggio [111]. Using a range of studies in psychology as well as eye physiology in humans and other animals they proposed a full sequence of steps for extracting depth from a pair of images including two tests for verification. The first rule is that each item in an image (area patch or line feature etc.) can only be matched against one item in the second image and the same in reverse, this is called the uniqueness test. Secondly the assumption that surfaces are smooth is used as described in the previous section, this is called the continuity test.

The usefulness of these tests will depend on the eventual use of the stereo data and what is more important accuracy or coverage. Like any refinement filter there will be some cases where good or “good enough” data fails to verify and is dismissed making the depth map more sparse. In the case of scene classification, it is most important to have as much coverage as possible as only a rough representation of a surface is needed as noise can be tolerated. In the case of 3D reconstruction noise would be highly disruptive to the appearance of the model making verification steps more important.

2.1.4 Distance calculation

Figure 2.1 shows a generic binocular camera configuration. The optical axes (Z_L and Z_R) are parallel and are separated by the baseline distance denoted as b in the figure (the physical distance between the sensors).

The definition of the stereo disparity is shown on the representation of the right-side image plane. It is the distance in pixels between the apparent position of a physical point in the left image and the apparent position in the right image. The further the object is from the cameras the smaller this distance will be.

In the stereo vision problem the goal is to determine the real-world distance of point P given the disparity d which was determined by finding projected positions P_L and P_R using a matching strategy discussed in the previous section. Using the pinhole camera model [15] this distance can be calculated from the baseline, b , camera resolution in pixels/mm, r , focal length in mm, f and the disparity d as follows.

$$\text{distance, } g = \frac{bfr}{d} \quad (2.1)$$

The actual x, y, z location of the point P can be recovered using the homogeneous perspective transform:

$$\begin{bmatrix} x' & y' & z' & W \end{bmatrix}^T = Q \begin{bmatrix} P_{Lx} & P_{Ly} & g & 1 \end{bmatrix}^T \quad (2.2)$$

where x', y', z' are intermediate values, W is the scale factor or homogeneous coordinate, P_{Lx} and P_{Ly} are the co-ordinates of the projection of P on the image plane I_L (Figure 2.1) and Q is the perspective transform matrix which allows adjustment for the viewer's position relative to the projection surface:

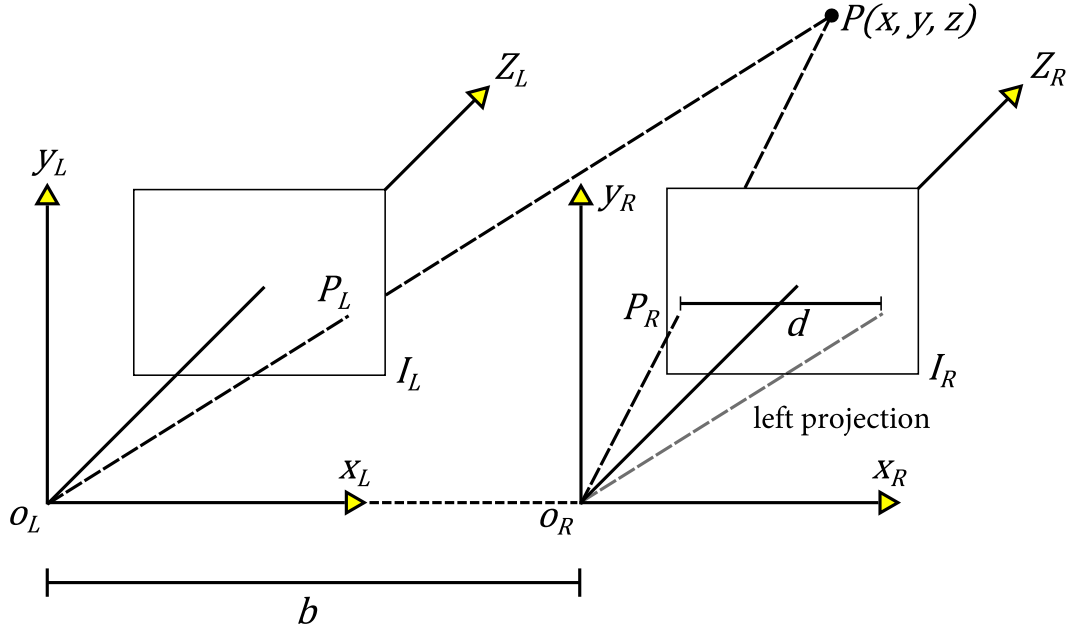


Figure 2.1: Binocular stereo vision geometry. Shows the stereo disparity d defined as the shift in projected position of real-world point P . The axes $X_L - Y_L - Z_L$ and $X_R - Y_R - Z_R$ represent the left and right cameras respectively and b is the baseline or the physical distance between the cameras as mounted. The positions P_L and P_R are the projections of point P on the left and right image planes, I_L and I_R .

$$Q = \begin{bmatrix} 1 & 0 & -\frac{e_x}{e_z} & 0 \\ 0 & 1 & -\frac{e_y}{e_z} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{e_z} & 0 \end{bmatrix} \quad (2.3)$$

where e_x, e_y, e_z are the viewer's position relative to the surface, the values 0, 0, 1 are used when this does not apply. The matrix is normally produced using the intrinsic properties of the camera used, see [Chapter 3](#) for a method of finding this matrix. Finally the actual x, y, z values are given using the perspective divide:

$$\begin{bmatrix} x & y & z \end{bmatrix}^T = \frac{1}{W} \begin{bmatrix} x' & y' & z' \end{bmatrix}^T \quad (2.4)$$

2.2 LIDAR TO CAMERA CALIBRATION

Before the sensors can be used in any fusion application there is an important intermediate step; calibration of the sensors. To map Light Detection And Ranging (LIDAR) measurements to the corresponding areas in the camera image accurately the precise translational and angular offset between the two devices must be known. This is a generic problem in sensor fusion; multiple information sources are useless unless the data from them can be aligned.

The process of converting the LIDAR measurements in real world 3D space to a location on the 2D image requires a co-ordinate system transform using these translational and angular offsets followed by a perspective transform requiring information about the camera's field of view and dimensions of the projection plane. Although these parameters can be easily measured from the physical installation or planned precisely before mounting it is the nature of mechanical fixings that there will be unexpected misalignment (and possibly movement after mounting), even careful measurements are unlikely to produce precise correspondence when applied to the data.

Previous works on calibration (Abidi and Chandra [1], Fishier and Bolles [52], Horaud et al. [79] and Quan and Lan [151]) have dealt with pose estimation of a camera using known points on an object in the image as reference points. This led to the general Perspective-n-Point (PnP) problem definition which is used to calibrate a camera relative to its environment using n 3D-to-2D point correspondences. Many solutions have been proposed to this problem (Zhang and Pless [193], Mei and Rives [119], Gao et al. [59] and Wasielewski and Strauss [184]) some of which consider a fixed number of points (P3P for example) used to create a system of equations to be solved using a non-iterative method. Others (Araujo et al. [10], Lu et al. [107] and Oberkampff et al. [134]) have considered an arbitrary number of point correspondences and iterative techniques to minimise some error function.

In support of optimisation methods in general Moreno-Noguer et al. [125] and Schweighofer and Pinz [159] present a comparison of different solutions to the PnP problem; they both conclude that approaches utilising error minimisation produced the most accurate calibration parameter estimates. However many of these methods assume the camera's internal calibration parameters are given, some require multiple poses of the camera to be captured for comparison and in the case of the iterative methods can take a considerable time to calculate. Another problem with the non-iterative methods is that in order to reduce calculation time some of them will approximate the error function used to determine the accuracy of the calibration, as a consequence the precision of the results can suffer.

2.3 MACHINE LEARNING

2.3.1 *Artificial neural networks*

An artificial neural network is a model fitting method inspired by the operation of biological neural networks in animals. The purpose of this is to solve problems that traditional linear computing cannot. The artificial neural network can be trained using a set of input and desired output data so that any future, unknown, input will be processed similarly to previously seen examples without being specifically pre-programmed to accept it. This is useful when input data contains noise which is not easily characterised but has common identifiable features which taken together indicate the required outcome. Artificial Neural Network (ANN)s are particularly well suited to pattern identification and have been successful in applications such as speech recognition [163].

In biology a neural network consists of many neurons (nerve cells) which are basic signalling units with many inputs and a single output. The input and output of these neurons may be from sensory organs or motor nerves respectively or other neurons in

the network depending on the neuron's position. Without going into unnecessary detail, each input to the neuron connects through a synapse which essentially controls the gain of the signal from each source.

In the mathematical model these synapses are modelled as an input “weight” which scales the signal by some fraction. Input weights can be positive or negative to represent excitatory or inhibitory connections and are generally between -1 and 1 . To determine the output of the node (artificial neuron) all the inputs are multiplied by their respective weights and summed together, this result is then fed into an activation function which sets the output. [Figure 2.2](#) shows a complete artificial neuron.

The activation function can vary depending on the application but at its most basic it takes the average of the weighted inputs, which should be in the range -1 to 1 , and assigns the output to either 0 (if average input ≤ 0) or 1 (if average input > 0).

The functionality of the [ANN](#) is mostly dependent on how the pool of nodes are connected together. The two main topologies are feed-forward networks, where information can only flow from input to output via numerous intermediate nodes provided there are no feedback loops, and recurrent neural networks, where nodes can feed back to previous nodes in the chain. When there is feedback the network may settle to a stable state for a given input or it may exhibit dynamic behaviour that constitutes the output of the system. In either case the nodes can be classified to one of three groups; input nodes that take input from outside, hidden nodes that take input from input or other hidden nodes and output nodes which send information out of the system. Smagt and Van Der [\[163\]](#)

Neural networks are trained by adjusting the weights using known pairs of input and output values. When the network is first initialised all the weights are set to a small random value so when the first input is presented the network will produce some random output. This random output is compared with the desired output and an error value is calculated, the weights are then adjusted slightly to reduce the error.

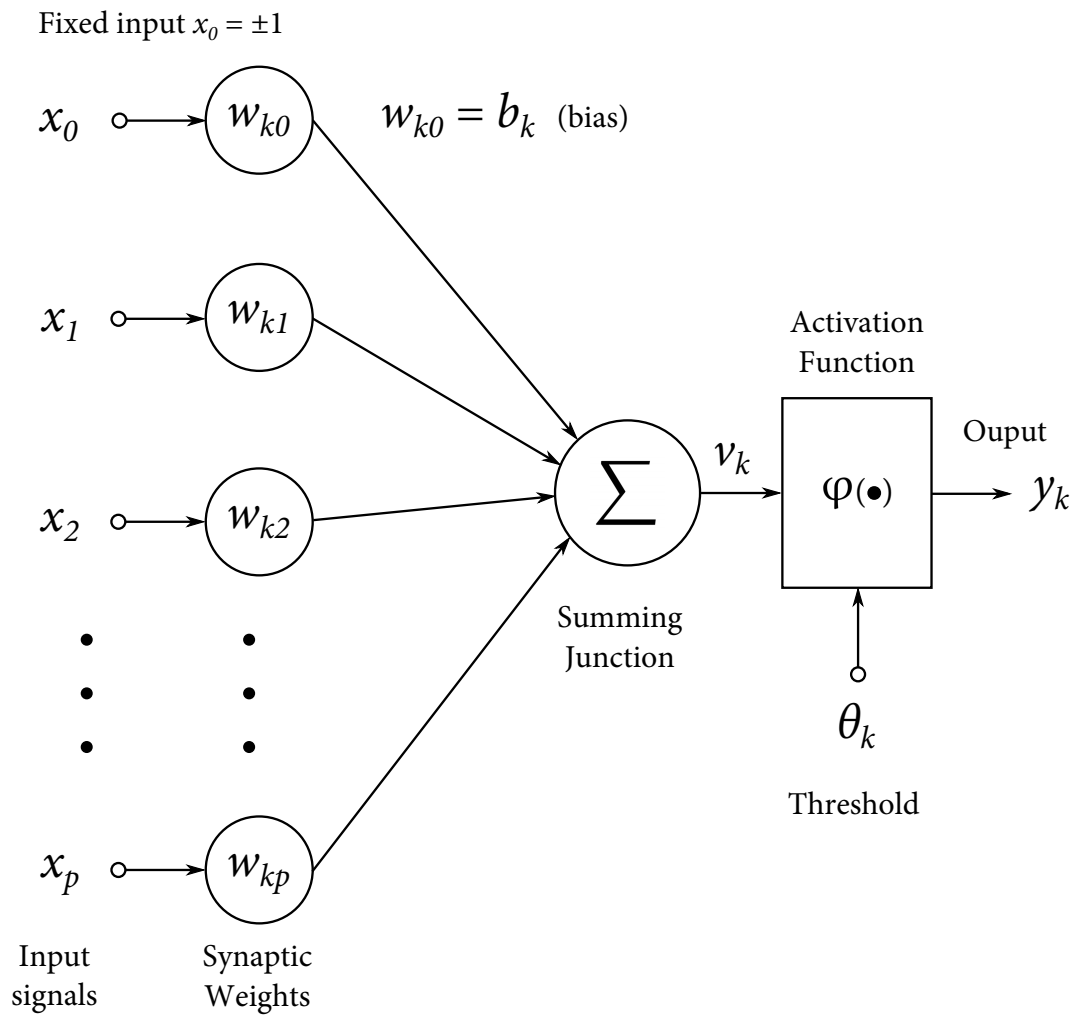


Figure 2.2: This demonstrates the operation of a classifying neural network, x are the input values, w are the node weights, v is the output value before clipping, φ is the activation function, generally a binary step function, θ is the threshold to trigger the activation function and y is the result. Recreated based on source: Smagt and Van Der [163].

The entire training set is presented to the network in this fashion and the error value for each sample is recorded. The node weights are then updated using the full error vector in a single operation determined by the training function. One pass through the training data is known as an epoch, often many epochs are processed before training is complete. Determining when training is complete is one of the difficult questions relating to neural networks. Given the great flexibility of the output signal there is a tendency for neural networks to be over-trained i.e. adjusting weights to specifically predict the training samples at the cost of losing the ability to deal with unseen samples. The problem will be discussed in [Chapter 6](#).

2.3.2 Support vector machines

The Support Vector Machine ([SVM](#)) is a widely used binary classifier which has gained popularity due to its ease of use and active development. The input data is generally presented as a set of normalised description vectors and a matching list of true or false class labels. A description vector typically consists of a list of observations of measurements on a particular item which needs classifying.

Given a set of description vectors as training data, \mathbf{x}_i , and a set of class labels, \mathbf{y}_i , where $\mathbf{x}_i \in \mathbb{R}^n$, $\mathbf{y} \in \{1, -1\}^l$ and $i = 1, \dots, l$ the [SVM](#) seeks to find the solution to an objective function. It can be written in many forms depending on the publication, here is a standard representation from Fan et al. [\[45\]](#).

$$\begin{aligned}
 & \min_{\mathbf{w}, \mathbf{b}, \xi} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^l \xi_i \\
 & \text{subject to} \quad \mathbf{y}_i (\mathbf{w}^T \phi(\mathbf{x}_i) + \mathbf{b}) \geq 1 - \xi_i, \\
 & \quad \quad \quad \xi_i \geq 0.
 \end{aligned} \tag{2.5}$$

Where C is the penalty parameter and is always greater than zero. The function ϕ in the objective function maps description vectors \mathbf{x}_i into a higher dimensional space such that a linear separating hyperplane with maximal margin between classes can exist in this space. \mathbf{w} is a set of normal vectors from each sample to the hyperplane, ξ is the loss function and b is the bias term.

This demonstrates the key benefit of [SVMs](#) - the training process by definition maximises the functional margin between classes which generally minimises the generalisation error. Also even if the classes are not linearly separable in their current set of dimensions and although the [SVM](#) creates a linear separation, the technique of mapping the samples into a higher dimension where a linearly separating hyperplane is more likely to exist allows non-linear classifications to be performed. The way samples are mapped or rearranged into this higher dimensional space is determined by the kernel function:

$$K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \quad (2.6)$$

Researchers have developed many varieties of kernel function however the most commonly used ones are [\[29\]](#):

- The linear kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- The polynomial kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i^T \mathbf{x}_j + r)^d$
- The Radial Basis Function ([RBF](#)) kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$
- The sigmoid kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i^T \mathbf{x}_j + r)$

Where γ , r , d are kernel tuning parameters and $\gamma > 0$.

It is interesting to note that an [SVM](#) using a sigmoid kernel is equivalent to a two-layer feed forward neural network showing that the two classifiers are closely related [\[35\]](#).

Based on the number of references in published articles, the [RBF](#) is by far the most popular choice. The reason most commonly given for this choice is that it only requires a single additional tuning parameter which makes it simple to determine the best pair of C and γ using a grid search. In addition the [RBF](#) always performs at least as well as the linear kernel because, as shown by Hsu et al. [81], the linear kernel is a special case of the [RBF](#). It also shows similar performance and classification characteristics as the sigmoid function for certain sets of tuning parameters as shown by Lin and Lin [102].

The sigmoid kernel is also commonly used however there is a known problem with the implementation where the function becomes invalid under certain conditions as noted by Cortes and Vapnik [35].

2.3.3 *K-means clustering for unsupervised learning*

Unsupervised learning takes a different approach to classification, instead of using existing examples of correct classifications to guide the separation of the classes it seeks to group samples together based only on their commonalities. Clustering is often used when there is an abundance of data but no existing model to separate it into groups. This opens up a different set of challenges to those encountered in supervised learning such determining the optimal number of classes which the data should be separated into. Also the challenge of deciding what defines two samples as being similar or having common properties. The ideal clustering algorithm will divide the samples in such a way that the similarity of samples within a group is maximised and the similarity of samples in different clusters is minimised.

One of the most common techniques to solve this problem in the literature is the k-means clustering algorithm. Its name refers to the parameter k which defines the number of classes the data should be grouped into and the application of means to determine the

similarity of a group. To determine the similarity of any two given samples the basic k-means algorithm takes the euclidean distance between the description vectors [108], the lower the distance the more similar they are considered to be. K-means has been used by others for image segmentation by grouping similar colours e.g. Marroquin and Giroi [112] and for grouping information such that, given one document, a group of similar documents can be retrieved e.g. Bellot and El-Bèze [18].

The basic procedure of the k-means starts by selecting k random points within the feature space (a space of n dimensions where n is the number of features in a sample description vector) as the initial cluster centres. Next, all samples in the dataset are assigned to their closest cluster centre using the euclidean norm distance. Once this is done the cluster centres are updated to be the mean of their current members. This is repeated until the sum of distances from each sample to its cluster centre converges on a minimum value [161, 165].

There are many variations on this method, usually relating to how the distances are calculated and how the initial centres are chosen. The best choice depends on the data being processed, the described method is the text book example. Bradley and Fayyad [20] shows improvements by choosing the initial cluster points based on simple tests of the input data. There are also modifications that allow the handling of different types of data, typically real numbers are used in the description vector but work by Huang [83] shows how categorical data can be incorporated and mixed with other data types to allow different ways of describing samples.

Wagstaff et al. [183], [182] attempts to improve the basic method by incorporating some prior knowledge if it exists. Although classification examples are not present it is sometimes the case that the researcher knows of fixed conditions that mean certain samples should never be in the same group or certain samples should always be in the same group.

Using these additional cues the work shows significant performance boosts over the entirely unguided method.

To address the problem of choosing k there has been extensive research, Fraley and Raftery [55] discusses the issue at length and covers several clustering algorithms in addition to k -means. Work by Pelleg and Moore [141] determines the optimal value of k by searching the data and optimising a Bayesian information measure. It shows significant improvement in average sample to cluster centre distances and at the same time improved computational efficiency. Tibshirani et al. [174] propose the use of a “gap statistic” which given the outcome of an initial k -means pass can estimate whether the chosen k is a good choice.

2.4 IMAGE PROCESSING

2.4.1 *Edge detection*

One of the most commonly used edge detection techniques is the Marr-Hildreth edge operator [110] which approximates the Laplacian of Gaussian $\nabla^2 G$ across the image and marks all points crossing zero in the convolution output as edges. Weak edges and noise may be reduced by using the magnitude of the gradient as it passes zero as a measure of edge strength, it is also possible to use this technique to filter out edges lying in a certain direction [154], since each edge point has an associated direction contour. Due to the flexibility of this approach it has been used by a number of authors (with small variations in its implementation) such as Kim and Aggarwal [89], Grimson [67, 68], Mayhew and Frisby [116], Pollard et al. [143] and Ayache and Faverjon [11], Ayache and Lustman [12] for primary image analysis and feature detection. Another method, based on the second derivative of the image, is the step-edge detector developed by Haralick [75] which offers better performance under certain conditions. For a comprehensive

performance comparison of these two methods see the work by Grimson and Hildreth [69].

First order derivative (gradient) filters such as Sobel [58] and Roberts and Prewitt [15] can also be used for feature detection. For example Baker and Binford [14] and Ohta and Kanade [135] used the gradient map to inspect the image one scan line at a time, in each scan line the peaks in gradient were used as features to be matched to the nearest peak in the opposite image. Again this method suffers from match points being sparsely distributed.

A major implementation using gradients is the Canny detector [25], this is a multi-step computational approach which provides a greater level of fine tuning in the operation, useful in situations where image processing parameters must adapt dynamically to constantly changing scenarios, such as in this research. The Canny method is sensitive to noise, so generally images are blurred using Gaussian filter, unless it is known that they will be noise free. Next the magnitude and orientation of the image gradient is calculated at all points using, for example, the Sobel filter [15]. Edges are then grown along strong saddle points in the gradient map tracing paths of minimal gradient variation i.e. non-maximal suppression using a dynamic hysteresis threshold. This threshold will depend on the edge strength of the local area allowing more subtle edges to be maintained provided they are connected to strong edges. This allows detection of softer edges without introducing false edges and noise into the output.

An improvement over the standard Canny algorithm, which is commonly used in field robotics, is the Canny-Deriche method [37]. The key advantage of this implementation is that it can be recursively computed making it suitable for running on embedded systems and Field-Programmable Gate Array (FPGA) chips commonly used on mobile platforms requiring vision processing.

2.4.2 Segmentation

The aim of image segmentation is to separate the image into a set of smaller regions often referred to as super-pixels or simply segments. It has become such an important preliminary step in so many computer vision related tasks that researchers have actually set up standardised segmentation performance tests such as the PASCAL challenge [43] and testing on the CALTECH dataset [47] encouraging ever better ways to divide images into meaningful parts.

The most basic segmentation approach, which does not consider image content, is a grid segmentation where the image is split into a uniform set of rectangular regions as used by Schroff et al. [157]. More commonly, however, it is desirable that the segments are chosen such that they define regions of similar content such as in the work by Hoiem et al. [77] which attempts to infer geometric properties from a single image then use these cues for segmentation.

Often the goal of obtaining segments, containing similar content, is to group them together and then identify what that common content actually is, a number of papers such as those by Brostow et al. [23], Shotton et al. [162], Wojek and Schiele [186] use visual and sometimes geometric information to identify the content.

As discussed above, edge detection can form the basis of a segmentation, but in general edge detectors do not aim to create closed regions within the image. However with some post processing of the edge maps this requirement can be satisfied. For example work by Christoudias et al. [32] introduces a method called EDISON which uses mean-shift clustering of the image data then uses edge detection to help split the regions. Mean-shift is effective at clustering regions of similar texture and ignores strong changes in intensity which is useful for grouping strongly textured areas like trees into a single region. The downside of this is that it often misses soft or weak edges in the image and combines

dissimilar regions into one. Edge detection on the other hand tends to over segment strongly textured areas.

2.5 GENERAL SEMANTIC CLASSIFICATION

The topic of semantic classification (also known as scene parsing, full-screen labelling) has its roots in mobile robotics where a system was needed to identify traversable terrain to aid in the process of autonomous path finding. This requires not only identifying different types of surface in the surroundings, but also understanding what restrictions those surfaces impose on navigation. The most basic type of environmental classification in mobile robotics uses only geometric features from a 2D [LIDAR](#) for example the work by Martinez-Mozos et al. [113], Buschka and Saffiotti [24], Anguelov et al. [8] and Limketkai et al. [101]. All of these papers use 2D [LIDAR](#) scans to identify important landmarks such as door frames and corridors which are useful navigational aids for indoor path finding. This is done by fitting line segments to the [LIDAR](#) scans and searching for the presence of gaps the width of a door in otherwise uninterrupted surfaces.

While doorways can be detected in this manner for more detailed understanding additional information about the surroundings is needed to not only improve detection accuracy but also to attach some kind of contextual information to the features detected. This will provide a more sophisticated understanding of the surroundings and allow better path planning to be made.

Some early work on semantic categorisation such as that by Schröter et al. [158] identifies common structural elements, again from 2D [LIDAR](#) maps and assigns them meaningful names. Similar techniques are used in SLAM map building by Kuipers and Beeson [92], Kuipers and Byun [93] to identify distinctive landmarks to use as navigational way-points.

The problem is that 2D scans alone do not possess sufficient information to robustly discriminate between true features and noise. The next step on from using 2D scans is to use a 3D scanning [LIDAR](#) which allows the detection of entire surfaces instead of just line segments. Using the additional information provided by vertical scanning, work by Nuechter et al. [133] shows that surfaces such as ceilings can be detected allowing mobile robots to avoid leaving the building for example.

With this additional 3D information it is now possible to take mobile robots into the outdoor world where the environment is significantly more unpredictable and unstructured. Work by Posner et al. [147] demonstrates the effectiveness of fitting planes to [LIDAR](#) data in an outdoor environment and shows that by using cues such as the height and orientation of a surface, relative to the ground, reasonable classification performance for floor and wall surfaces can be achieved. Even more relevant to [ADAS](#) systems is the work by Anguelov et al. [9] which uses the segmentation of 3D point clouds and [SVM](#) machine learning to detect the presence of road vehicles. Classification from 3D point clouds is not limited to driver-perspective scene segmentation. It has also been used on full 3D point clouds representing a building or larger geographic area to create an automatic map of surfaces. Xiong and Huber [189], [190] make particular use of the contextual information available in the 3D scan data. In the later paper clusters of points in the main point cloud are split apart and their relationship to each other is described using a number of positional cues. Then, using conditional random fields on the derived relationship data, a scene containing several buildings and forested areas is successfully labelled.

More recently with the availability of cheap cameras with reasonable performance many projects have been started to look at the incorporation of image data into this kind of classification. Image data is useful for differentiating between what are otherwise identical bumpy surfaces. The information is particularly useful for the problem mentioned above of identifying traversable terrain as demonstrated by Hadsell et al. [71]. In many cases the

combined systems discussed later have drawn inspiration from image only segmentation and categorisation techniques such as those presented by Ponce et al. [144] and Pope [145].

The logical progression from these two types of segmentation is to use them both together, the problem now becomes how to most effectively divide and categorise the large quantity of available data. One approach put forward by Douillard et al. [38] uses conditional random fields to identify vehicles on the road and claims to be able to incorporate any number of sensor inputs from both LIDAR and camera provided they are correctly calibrated. Another approach by Monteiro et al. [122], which, instead of combining all the data sources together, does independent classification using each sensor and then decides the final outcome with a Bayesian estimator. The work of Happold et al. [74] uses stereo vision instead of a 3D LIDAR to provide physical structure and combines this with neural network machine learning to classify terrain types but without any semantic information.

A paper authored by Posner et al. [148] shows a similar attempt to categorise different surface structures in the environment of a small mobile robot. Using a fully 3D scanning LIDAR and a single camera to provide some visual information the results shown in Figure 2.3 were obtained. The approach uses plane fitting on the geometric data to segment the space, then extracts matching image data for each plane. By combining the SURF features and some geometric features from the LIDAR such as goodness-of-plane-fit, and orientation, a bank of SVM classifiers are trained and used to categorise segments. The work shows promising results, but remains unsuitable for automotive applications due to reliance on 3D scanning LIDAR and high-dimension description vectors leading to slow classification performance. Similar work by Munoz et al. [128] uses a camera and a fully 3D scanner. The main focus in this paper is to resolve the problems that arise in scene labelling, which depends on two or more sensors, when all sensors cannot cover

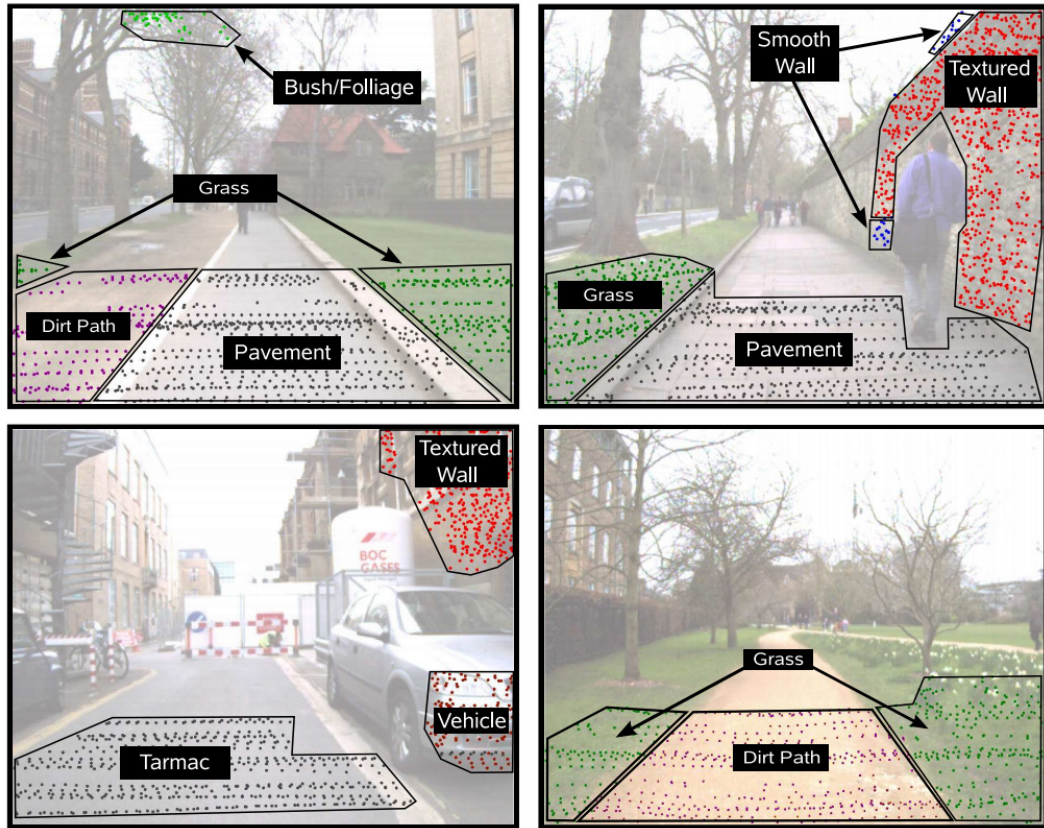


Figure 2.3: This is the demonstration figure produced by Posner et al. [148] in the paper *Online Generation of Scene Descriptions in Urban Environments*. It shows the LIDAR based segmentation and classification of regions.

the same parts of the scene. The paper describes and compares a number of inference techniques to fill in missing data from one sensor using known data from the others.

At other times it is not important to break an image up and determine what every pixel in an image represents, instead researchers are interested in determining what is generally depicted. For example classification software which can sort a group of images by items they contain e.g. a group containing people and a group containing dogs. These are often referred to as bag-of-words classifiers, two of the most well known examples of which can be found in the papers by Lazebnik et al. [97] and Fergus et al. [50].

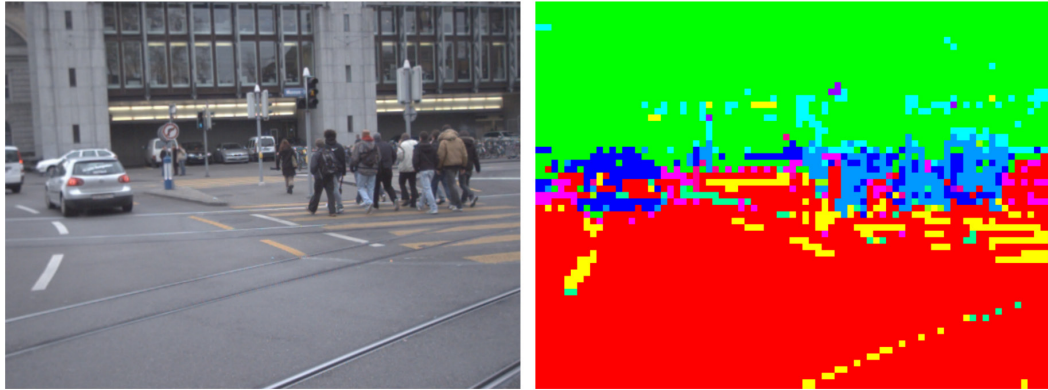


Figure 2.4: This is the demonstration figure produced by Ess et al. [42] in the paper *Segmentation-based urban traffic scene understanding*. It shows a patch based classification of a busy junction.

Some work combines these two different approaches and tries to classify the type of scene depicted while at the same time segmenting the image and classifying the content of each super-pixel for example the work by Cao and Fei-Fei [26].

In the more specific case of intelligent vehicles the technique in the paper by Oliva and Torralba [136] is specifically designed to distinguish between different types of driving environments such as highways, smaller streets and rural areas. This kind of work could be useful for switching between different sets of image segmenters and segment classifiers depending on the driving environment. Another example by Ess et al. [42] attempts this by simultaneously classifying driving environment and image contents as seen in Figure 2.4. The work uses AdaBoost learning (described in Freund and Schapire [56], Tieu and Viola [176]) on SURF features to learn the appearance of different road markings, objects and ground surfaces.

The method shows good results on road marking classification but does tends to confuse objects such as pedestrians and vehicles. Detecting pedestrians reliably is one of the most sought after techniques for car manufacturers as having a reliable system would allow effective avoidance of pedestrian collisions even if the driver is not paying attention. Work

by Ess et al. [41] and Gavrilu and Munder [61] attempts to detect them more robustly by tracking moving segments in the image and using motion cues to aid classification.

Scene parsing is a growing topic of interest, as such researchers are keen to directly compare new methods to determine the most promising approaches. The best way to do this is to use established set of images with known label maps such that performance can be accurately measured. Some examples of existing datasets are the *Stanford background dataset* [66] the *LabelMe dataset* [155]. Unfortunately the currently available image sets are not limited to road scenery, and they do not include stereo pairs for depth extraction. For this reason it was not possible to directly compare results later in this thesis using these sets.

In the more general context of scene parsing using the mentioned datasets some promising approaches have been but forward using techniques other than machine learning. For example Liu et al. [104], [105] and Tighe and Lazebnik [177] attempt to label images via direct comparison to images with known labelling. First the unknown image is matched to the most similar labelled image in the database then statistical segment matching is used to infer the correct labelling. Performance is promising on a subset of the *LabelMe dataset*, however currently the technique does not make use of spacial information.

Another general scene labelling technique by Farabet et al. [46] aims to analyse a wide variety of images to find specific surfaces such as mountains, fields, windows, building, cars, doors, river. The technique inspects each pixel in the image with a dynamically assigned neighbourhood. Initial predictions from a segment classifier are refined using the optimal path through a labelling tree to minimise the entropy of class distribution. This method claims to have the best performance on the *sift flow* image dataset but again it is aimed at general images and does not use spacial information, meaning a comparison is not possible.

3

LIDAR CALIBRATION AND FUSION FOR IMPROVED DEPTH MAPS

CONTENTS

3.1	Calibration overview	48
3.2	Problem definition	49
3.3	Approach	52
3.3.1	Nelder-Mead algorithm	52
3.3.2	Definition of the objective function	53
3.4	Previous work to determine number of required targets	58
3.5	Collection of test data	61
3.6	Results and discussion	68
3.6.1	Calibration results	68
3.6.2	Validation of results using initial conditions	70
3.6.3	Validation using independent samples	72
3.7	Fusion and merging of sensor data	72
3.7.1	Final calibration	74
3.7.2	LIDAR interpolation	75
3.8	Theoretical stereo noise performance	79

3.1 CALIBRATION OVERVIEW

In this section the Nelder-Mead simplex search method (see [Section 3.3.1](#) on page 52 [94]) is used to obtain a set of optimised calibration parameters (external offsets and internal camera parameters) by minimising the alignment errors between the reference image co-ordinates and the re-projected lasers scanner data. In comparison with other approaches, this approach has the following benefits:

1. This method is able to calibrate both internal and external calibration parameters simultaneously without the need for separate stages. This is because the error function, defined for optimisation, can incorporate various types of parameter including co-ordinate transformation and internal camera parameters such as a distortion function.
2. Instead of using multiple poses of the same object, and time consuming test scenarios using specific test patterns required by some approaches, this method requires only a small group of easily made targets that are spread across the sensor's Field Of View (FOV). They can then be captured in a single pose which provides all the data necessary to calibrate.
3. This is the only calibration method that does not require the camera image to be pre-rectified.
4. This approach employs an iterative process to achieve an optimised estimation. The results can be successively improved by applying multiple passes to address the problem of convergence on non-global minima which is common to many iterative optimisation approaches. Each pass uses the solution from a previous run to seed the next, each time producing a better estimation [140]. This does not guarantee that the global minimum will be found but it has been shown to provide a significant improvement in this particular case.

This section is a continuation of the work published in the *IEEE 2010 Intelligent Vehicle Symposium* [140] and was later published in *Measurement science and Technology* [139].

3.2 PROBLEM DEFINITION

The physical setup of the sensor system is represented by the model shown in Figure 3.1. In this model there are twelve parameters, $(C_x, C_y, C_z, \theta_x, \theta_y, \theta_z, u_0, v_0, f, k_1, k_2, k_3)$ necessary to describe the transformation between the laser scanner co-ordinate and image plane co-ordinate systems. The objective of this work is to determine values for these twelve parameters such that the error between known pixel locations and pixel locations generated using these values is minimised. All parameters except f, k_1, k_2, k_3 are illustrated in Figure 3.1. The parameter f is introduced to scale the perspective transformed data points onto the image plane, it encapsulates the camera's FOV and the resolution of the image. In this case it is defined as half width of the image in pixels divided by the tangent of half the camera's horizontal FOV. It can also be defined using the focal length of the lens, depending on what information is available to the operator, since this value is related to the FOV by the physical size of the camera's Charge-Coupled Device (CCD). This information is provided by the manufacturer but since any manufactured goods are subject to built tolerances and assembly variation it cannot be assumed that the values are exactly correct. The distortion coefficients k_1, k_2, k_3 are used in the generally accepted barrel distortion model [5]:

$$\begin{pmatrix} u^* - u_c \\ v^* - v_c \end{pmatrix} = L(r) \begin{pmatrix} u - u_c \\ v - v_c \end{pmatrix} \quad (3.1)$$

$$r = \sqrt{(u - u_c)^2 + (v - v_c)^2} \quad (3.1a)$$

$$L(r) = 1 + k_1 r + k_2 r^2 + k_3 r^3 \quad (3.1b)$$

where:

- $L(r)$ distortion function.
- u^*, v^* distorted projection points.
- u_c, v_c centre of distortion (assumed to be centre of image).
- u, v undistorted 2D projected points.
- r distance from the centre of the image.
- k_1, k_2, k_3 1st, 2nd and 3rd order distortion coefficients.

This function takes the projected laser scanner data (after rotation and perspective transformation) and deforms it according to the distortion model and the estimated distortion coefficients. Since the data is to be transformed from the 3D laser scanner co-ordinate system to a point on the 2D image plane, the laser scanner co-ordinate system is taken as the base reference frame therefore the x, y, z offsets in real space, represented by vector (C_x, C_y, C_z) are all relative to the origin of the laser scanner measurement space, O_L . The same applies to the rotation vector $(\theta_x, \theta_y, \theta_z)$. All angles are relative to the orientation of the laser scanner.

In most cases it is desirable to have the camera as high as possible to get an unobstructed view of the scene ahead and to have the laser as low as possible so that it can scan parallel to the ground without missing low obstructions. Due to these constraints it is expected

that C_y will be relatively large. In addition, since the camera is some distance from the floor and has a limited vertical field of view, it will not be able to observe objects close to the laser scanner unless it is tilted towards the ground. Therefore it is expected that θ_x will have some significant non-zero value as shown in Figure 3.1.

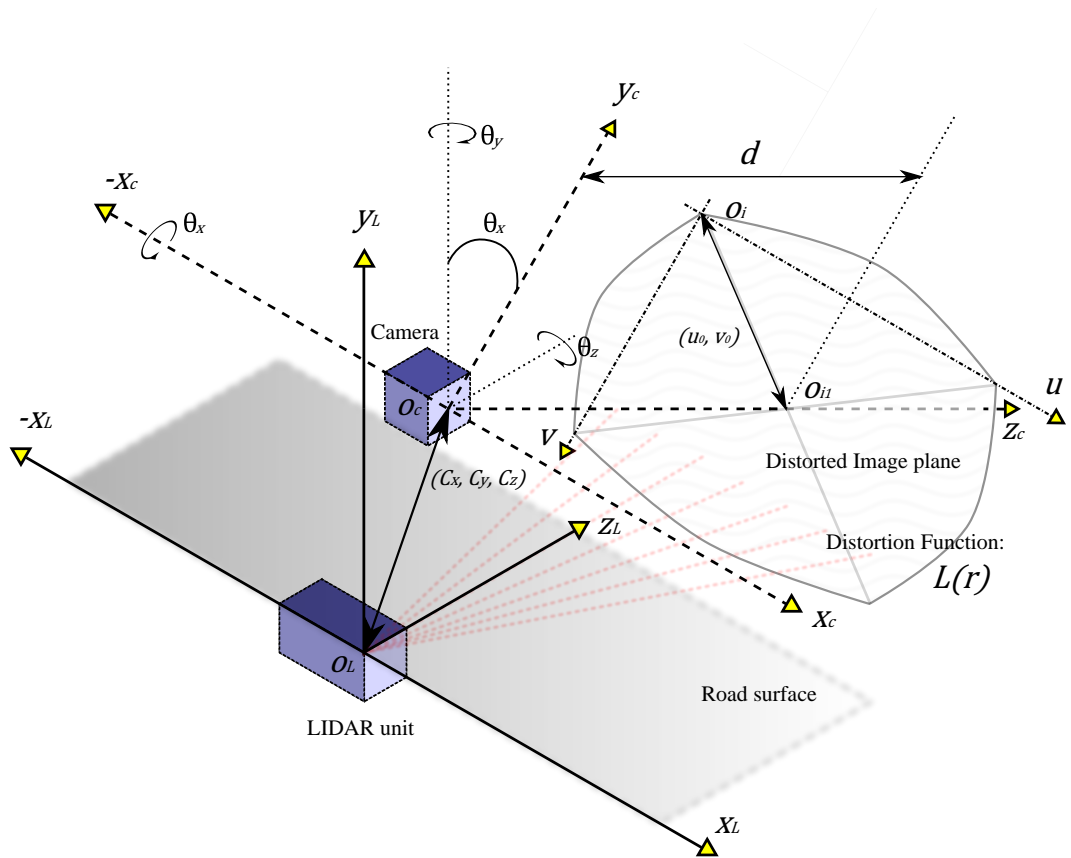


Figure 3.1: Diagram showing the offsets between the two non-conventional co-ordinate systems (not to scale), the $X_L - Y_L - Z_L$ axes with origin O_L is the laser scanner, or World co-ordinate system, the $X_C - Y_C - Z_C$ axes with origin O_C is the camera co-ordinate system and the $u - v$ axes with origin O_i is the image co-ordinate system. The point O_{i1} is the optical centre of the image plane where it is intersected by the optical axis Z_C . The vector (C_x, C_y, C_z) is the (x, y, z) translational offset from the laser scanner to the camera, the vector $(\theta_x, \theta_y, \theta_z)$ is the (x, y, z) rotation between the reference systems. The scalars u_0 and v_0 are coordinates of the optical centre in pixels in the image coordinate system. Dashed lines coming from laser scanner unit illustrate the path of the laser pulses.

After the co-ordinate transformation from world space, to camera space, a perspective transform is applied to project the transformed point onto the image plane. Using the usual pinhole camera model this means there will be a single vanishing point in the projection, where all objects will converge, as the projected point's distance d from the observer tends to infinity. This point is at the intersection between the camera's optical axis, Z_c , and the image plane, ideally it should be in the centre of the image. However due to manufacturing error it may deviate unexpectedly. To compensate for this error scalars u_0 and v_0 are used to define the true co-ordinates of the camera's optical centre, O_{i1} , in image co-ordinate system ($u - v$ co-ordinates).

Among the twelve parameters, the first six are external factors indicating the mounting position relationship between camera co-ordinate system and laser co-ordinate system. The last six are camera internal parameters indicating camera manufacturing specification since all cameras will have varying degrees of error in these values it is necessary to consider them as variable. For reference, the image size used in this work is 640 by 480 pixels.

3.3 APPROACH

3.3.1 *Nelder-Mead algorithm*

The Nelder-Mead simplex direct search algorithm is a type of unconstrained non-linear optimisation. A full description of the algorithm can be found in this work by Lagarias et al. [94] however a brief description is given here. The implementation used in this work is provided by the *fminsearch* function within Matlab. The algorithm defines an objective function to optimise a vector of input arguments. In this case the input vector has 12 elements as defined previously. It also requires an initial estimate of the input arguments. Essentially the algorithm begins by defining a simplex (the analogy of a

triangle in p -dimensional space) of $p + 1$ points where p is the number of parameters to be optimised. The i^{th} vertex in the simplex is given by adding 5% of each element in $x_0(i)$ to the corresponding element in the original x_0 such that each vertex is the same as the original x_0 except a single respective element which is increased by 5%. The final vertex $p + 1$, is given by x_0 itself, such that the simplex is centred about the initial estimate. Each vertex in the simplex represents a possible solution when used as an input to the objective function. During each iteration of the algorithm each of the vertices is used in the objective function and evaluated to calculate the associated error at each vertex. The results are arranged in order and generally the vertex with the highest error is moved to a new point determined by the operation performed in the current iteration (although some operations replace most or all of the vertices). The operation performed during the iteration is decided using the result of tests involving the error measured at intermediate points within the simplex. If a better solution is found, this replaces the worst solution from the previous iterations. Possible operations are Reflect, Expand, Contract outside, Contract inside, and Shrink as explained in the paper by Lagarias et al. [94]. The algorithm continues this process of refining the solution until a solution is found that gives an error value of zero (or reduced to within a pre-defined threshold) or until convergence is reached i.e. no further improvement can be made for continuing iterations.

3.3.2 Definition of the objective function

Using the parameters identified the problem can be expressed in the form;

$$\varepsilon = \phi(C_x, C_y, C_z, \theta_x, \theta_y, \theta_z, u_0, v_0, f, k_1, k_2, k_3) \quad (3.2)$$

Where ε is a scalar and a function of the twelve parameters. The objective function, ϕ , must incorporate the effects of all the parameters when applied to some provided observation data and return a single scalar representing the error. The code for this function can be found in [Appendix A](#) on page 226. The Nelder-Mead algorithm then adjusts the inputs until the error, ε , is minimised. The objective function is defined as follows;

$$\varepsilon = \sum_{i=1}^n ((\mathbf{O}_{u,i} - \mathbf{N}_{u,i})^2 + (\mathbf{O}_{v,i} - \mathbf{N}_{v,i})^2) \quad (3.3)$$

where:

- ε scalar representing the calibration error
- n the number of sample points (corresponding observations)
- \mathbf{O} a 2 by n matrix containing image co-ordinates of all sample points measured by the camera, each column is a $u - v$ vector corresponding to the matching column in \mathbf{L} . $\mathbf{O}_{u,i}$ is the i^{th} element of row u of \mathbf{O} .
- \mathbf{L} a 3 by n matrix containing world space co-ordinates of all sample points measured by the laser scanner, each column is a three element position vector. Rows ordered x, y, z .
- \mathbf{D} a 3 by n matrix which is the co-ordinate transform of \mathbf{L} into camera space i.e. an intermediate stage between \mathbf{L} and the 2D projection \mathbf{Q} . Rows are ordered x, y, z .
- \mathbf{Q} a 2 by n matrix containing all the newly calculated $u - v$ pixel co-ordinates after applying perspective transform to \mathbf{L} using the given input parameters i.e. the 2D projection of \mathbf{L} . $\mathbf{Q}_{(u,i)}$ is the i^{th} element of row u of \mathbf{Q} .

\mathbf{N} a 2 by n matrix containing all the $u - v$ pixel co-ordinates after applying barrel distortion to \mathbf{Q} using the three distortion coefficients i.e. the 2D distorted projection of \mathbf{L} . $\mathbf{N}_{u,i}$ is the i^{th} element of row u of \mathbf{N} .

The function calculates the sum of squares of the Euclidean distances between each point pair in \mathbf{N} and \mathbf{O} which represents their degree of alignment. Within the body of the function, the values of all of the parameters are supplied by the current iteration in the Nelder-Mead algorithm.

The transformation from \mathbf{L} to \mathbf{N} has three steps; the first step is the rigid co-ordinate transformation from the world space \mathbf{L} , to the camera space \mathbf{D} . The transformation can be expressed as:

$$\mathbf{D} = \mathbf{T} \cdot (\mathbf{L} - \mathbf{C}_E) \quad (3.4)$$

Where \mathbf{C}_E is a 3 by n offset matrix where each column is a copy of the vector (C_x, C_y, C_z) , and

$$\mathbf{T} = \mathbf{R}_x \cdot \mathbf{R}_y \cdot \mathbf{R}_z \quad (3.4a)$$

Where $\mathbf{R}_x, \mathbf{R}_y, \mathbf{R}_z$ are co-ordinate rotations in three dimensions and can be written

$$\begin{aligned}
\mathbf{R}_x &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-\theta_x) & \sin(-\theta_x) \\ 0 & -\sin(-\theta_x) & \cos(-\theta_x) \end{bmatrix} \\
\mathbf{R}_y &= \begin{bmatrix} \cos(-\theta_y) & 0 & -\sin(-\theta_y) \\ 0 & 1 & 0 \\ \sin(-\theta_y) & 0 & \cos(-\theta_y) \end{bmatrix} \\
\mathbf{R}_z &= \begin{bmatrix} \cos(-\theta_z) & \sin(-\theta_z) & 0 \\ -\sin(-\theta_z) & \cos(-\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{3.4b}$$

This form of the co-ordinate transform equation assumes a left-handed co-ordinate system [126] as shown in Figure 3.1. The second step is the perspective transform [27] from \mathbf{D} to \mathbf{Q} expressed as:

$$\mathbf{Q}_u = f \frac{\mathbf{D}_x}{\mathbf{D}_z} + u_0 \tag{3.5}$$

$$\mathbf{Q}_v = f \frac{\mathbf{D}_y}{\mathbf{D}_z} + v_0 \tag{3.6}$$

\mathbf{D}_x , \mathbf{D}_y , \mathbf{D}_z are rows 1, 2 and 3 of matrix \mathbf{D} , the division used in Equation 3.5 and Equation 3.6 is element-wise division *not* matrix division (i.e. each element of the numerator is divided by the corresponding element in the denominator and the result is a vector of the same size). Similarly the addition of u_0 , v_0 and multiplication by f is also applied element-wise.

The third and final step is to apply the lens distortion model to the projected points in order to match the distortion observed in the camera image. This is applied using the model defined by Equation 3.1, Equation 3.1a and Equation 3.1b. The variable names used in the distortion model do not match the variable names defined in this section, to make the equations applicable the following substitutions are made: inputs u, v will become Q_u, Q_v , outputs u^*, v^* will become N_u, N_v and the distortion centre u_c, v_c will be set to 320, 240 i.e. the centre of the image.

The flow of information through the algorithm

In summary the following steps describe the work flow when using this method to calibrate a fusion system:

1. Take a snapshot of a scene containing objects of known size and location which are visible to both the camera and laser scanner.
2. Record a number of points, n , in the image that have known corresponding data points in the laser scanner profile; these are the \mathbf{O} and \mathbf{L} sample data matrices.
3. Take measurements of the physical offsets between the sensors and determine estimates for the other parameters.
4. Run the Nelder-Mead simplex direct search algorithm using the objective function Equation 3.3 and the approximate physical offsets as initial estimates of the parameter set. The search will make improvements to all parameters over many iterations until the error, ϵ , is within a predefined threshold or a convergence has been reached.
5. If solution is unsatisfactory the first pass solution may be reused as the initial estimate for an additional optimisation pass by repeating step 4. This can be done

repeatedly until convergence is reached (i.e. no further change in solution) or until the error magnitude is acceptable.

3.4 PREVIOUS WORK TO DETERMINE NUMBER OF REQUIRED TARGETS

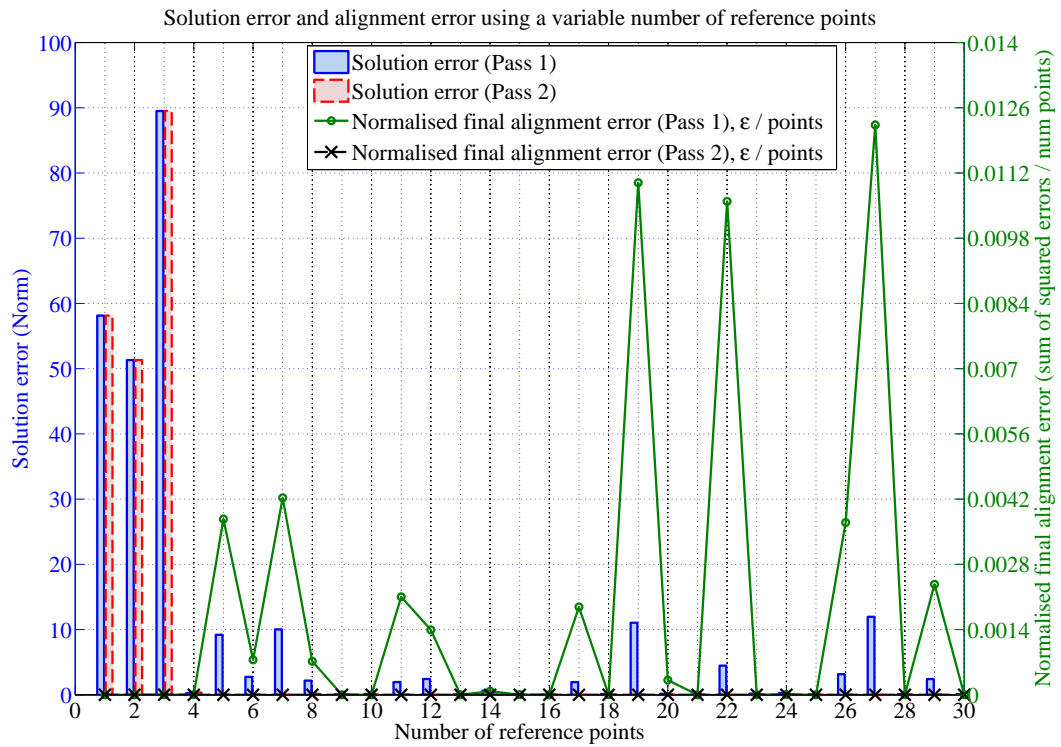


Figure 3.2: Plot of results from using increasing numbers of input data points, the error and deviation from true solution at the completion of each optimisation are shown.

The preliminary work was an experiment to prove the concept that calibration parameters could be determined using only raw Light Detection And Ranging (LIDAR) data and intended image projection points. This was carried out using a set of synthetic data to allow quick and accurate testing free from the effects of human, or measurement, error.

The data set for these tests was generated by taking an entire 978 point data frame from the [LIDAR](#) (used for L , see [Equation 3.4](#) on page 55) and aligning it to an image (captured simultaneously to the [LIDAR](#) data) by manually adjusting the parameters to get the best visual fit. The parameters chosen after manual alignment were used to transform the [LIDAR](#) data L and the result was taken as the corresponding pixel locations, O .

Using the data, the proposed method was used on increasingly large subsets of corresponding points to determine the minimum number of points required to get a robust fit. [Figure 3.2](#) shows the results of this experiment. To produce this graph the proposed method was run multiple times using an increasing number of (spatially well distributed) reference points from 1 to 30 instead of the entire 978 point dataset. Each run began with the same initial estimate and ran for two passes.

Looking first of all at the solution error, after the first pass, (blue bar, left axis) it can be seen, as expected, that all experiments with a small number of data points (less than 4 in this case) have an extremely large error, indicating that it was not possible to identify the true solution correctly. Interestingly, the alignment error in this region, (less than 4 input points) is zero indicating that the optimisation algorithm successfully manipulated the parameters to align the data points precisely to get zero error but the parameters used to achieve this result were not representative of the physical setup. This is an issue to be aware of when interpreting the results: if the input data is not well chosen then the algorithm may converge on a solution that fits only the specific data given but won't be appropriate in the general case. Well chosen in this instance means well distributed across the FOV of the sensors and well distributed across the range of observable distances, tightly clustered points will not provide sufficient constraints on the optimisation. As soon as the number of data points increases past 4 the solution error drops down with correlated errors in final alignment error. Many points in this region (15 points) end up with almost zero solution error; however in the cases where error is present it is generally

large. This is caused by the existence of a local minima near the optimal solution which are occasionally encountered on the first pass in this particular data set.

The solution error after a second pass (red bars). For the first 4 points the solution is identical to the first pass i.e. no further optimisation is possible. However for the points between 5 and 30 the result is consistently near zero i.e. almost perfect. The final normalised alignment error on the second pass is less than $10^{-5} \epsilon$ / points across the entire range. The solution for the runs with 3 or fewer data points cannot be improved since there are insufficient constraints on the objective function. For this reason the second pass has no effect on the first 3 points as the objective function is already minimised to zero using incorrect parameters. However, for the rest of the runs, some of which converged on a local minimum on the first run, the solution error was reduced from its previous value to near-zero. In the case of the points 19 and 27, examined above, the local minimum they converged on was sufficiently close to the global minimum such that on the second pass it was found, thus demonstrating the advantage of additional passes.

The main conclusion from this work [140] was that a minimum of 10 corresponding points are needed for reliable calibration and they must be distributed across the camera's FOV as far as is possible and cover a wide range of distance from the sensors.

The work described in the rest of this section is a practical implementation of the approach proposed in [140] by using sample data captured from an experimental scenario in the field. Detailed experimental design and process have been added. Furthermore, the algorithm is updated with the addition of a barrel distortion model which makes the approach completely applicable to real situations (see Equation 3.1 on page 49).

3.5 COLLECTION OF TEST DATA

In order to test the method a vehicle was setup with a camera on the roof at the back and a laser scanner under the licence plate such that both sensors are observing the same area as shown in [Figure 3.4](#). A total of 11 targets were placed to get corresponding laser scanner and image point pairs, each made from a small paper disk attached to a stick as shown in [Figure 3.3](#).



Figure 3.3: Target used for LIDAR calibration, it is constructed from a piece of bamboo and a cardboard disk cut to a specific diameter.

The size of the target is the most important consideration – it should occupy the smallest space possible on the camera image (ideally a single pixel) allowing maximum certainty in determining its position in the camera reference frame. The laser scanner should also detect these targets and ideally only register 1 point on the target making it easy to determine the exact corresponding world location. Determining its position precisely in the laser scanner frame is not possible as minimum measurement error will always be



Figure 3.4: Physical setup of the fusion system.

proportional to the laser scanner's angular step as it scans across and the distance from the laser scanner.

The size of the real world space corresponding to one pixel in the image is proportional to the distance from the camera. The radius (p_w) of the target can be calculated using the following equation:

$$p_w = \frac{2 \cdot d \cdot \tan\left(\frac{c_{fov}}{2}\right)}{640} \quad (3.7)$$

Where c_{fov} is the camera's horizontal field of view, d is the distance from the camera and 640 is the horizontal resolution of the camera in pixels.

The laser spacing can be calculated the same way, given that the laser scanner scans at intervals of 0.25° the gap between data points at distance d is simply:

$$d_{gap} = \tan(0.25^\circ) \cdot d \quad (3.8)$$

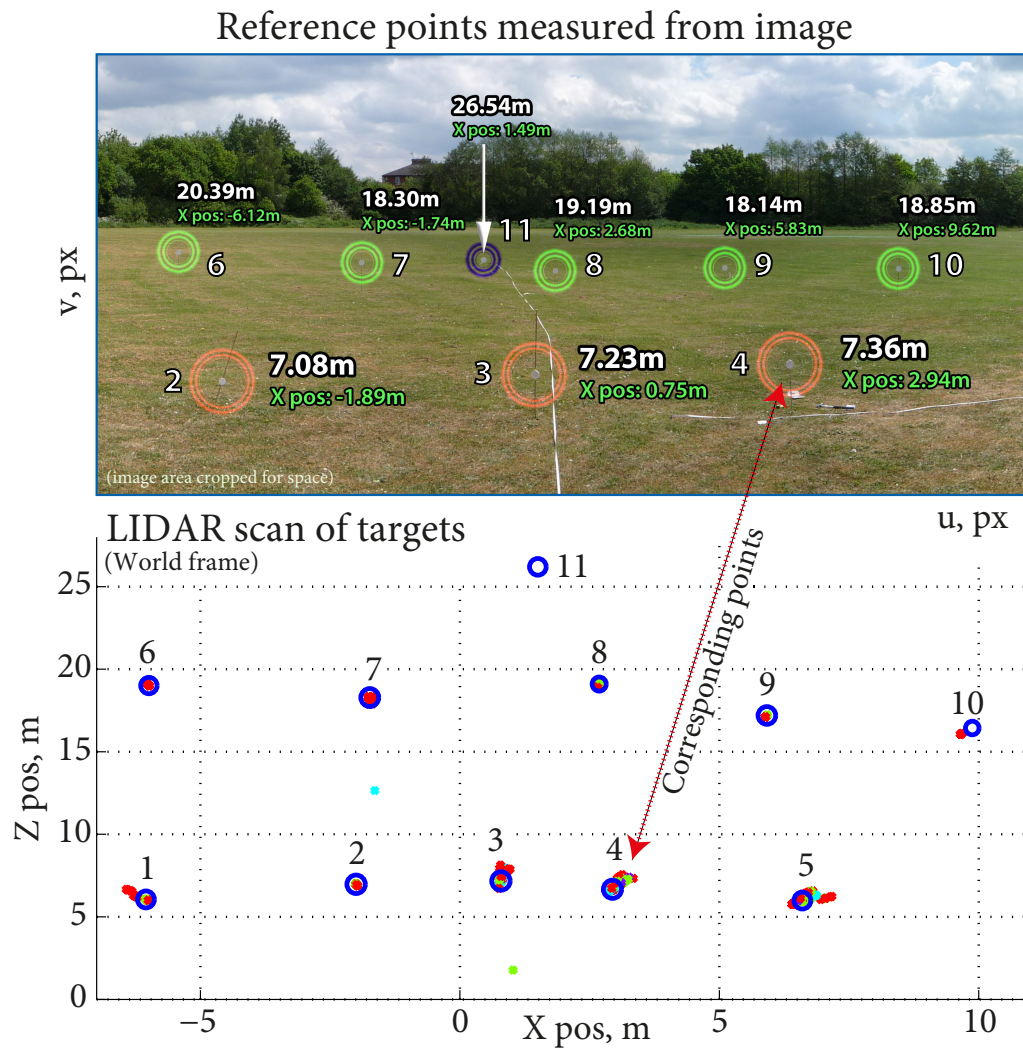


Figure 3.5: Captured data from the camera and the laser scanner observing the 11 targets, the laser scanner data is a real plot from the device. The corresponding points are matched up and assigned an ID from 1 to 11. Targets one and five are shown in the LIDAR scan but have been cropped from the camera image to avoid shrinking the figure until distant targets are not visible.

Figure 3.6 is a simple graph showing this relationship. It is important to note that the laser spacing i.e. the distance between consecutive measurement points on the laser scanner is always smaller than the width of a target. This relationship guarantees that a target made to the specified size calculated from Equation 3.7 will always intercept at least 1 laser point making it detectable by both devices.

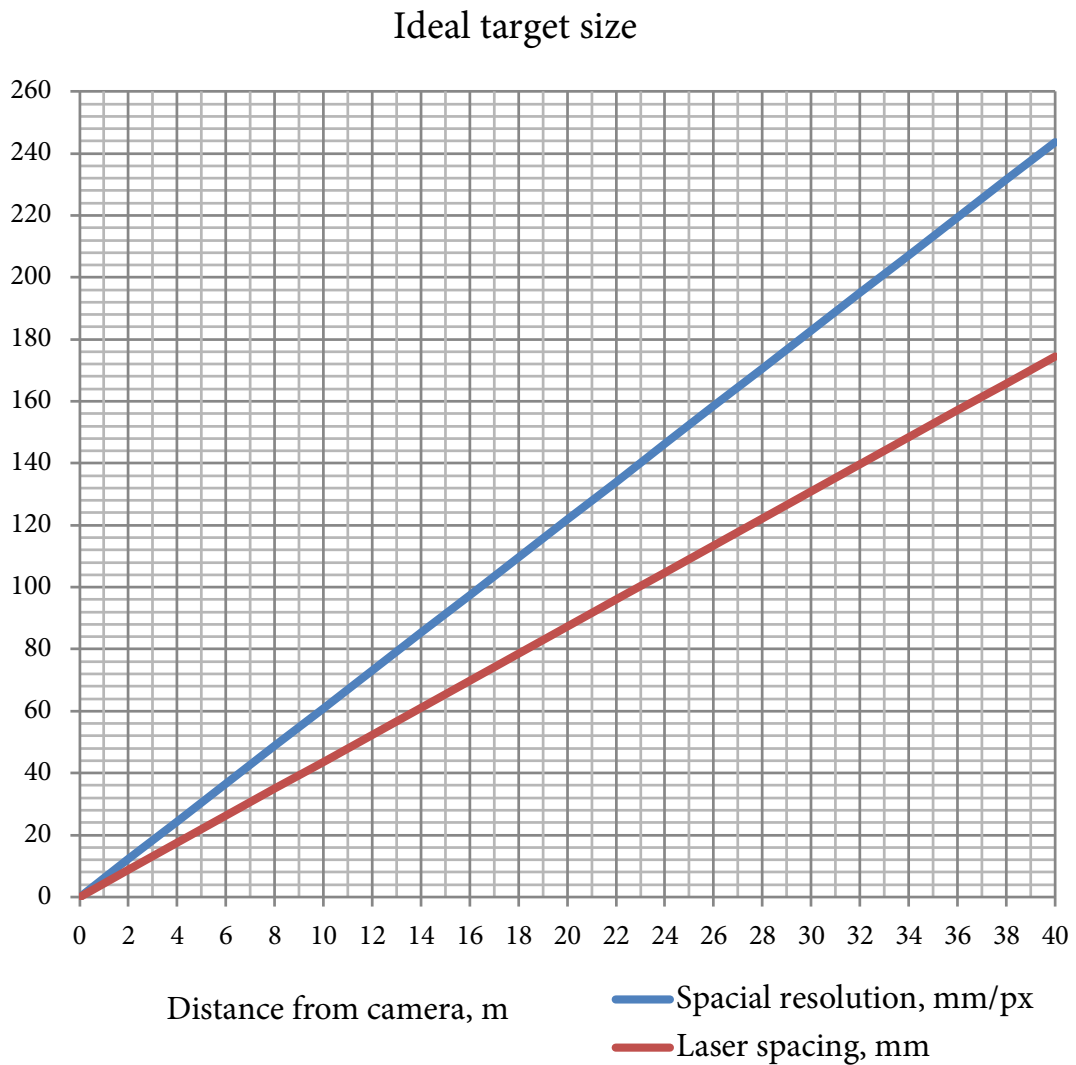


Figure 3.6: Graph to show required target size at a given distance such that it occupies a single pixel in the image and intercepts at least 1 laser point. Blue line is Equation 3.7, red line is Equation 3.8.

Another important factor to consider is the distribution of the target points. The previous paper Osgood and Huang [138] concluded that the targets must be well spread across the image width and height and also in terms of distance. Since the laser scanner is planar and only scans across a narrow horizontal band the only way to spread the targets across the height of the image is to move them closer to the vehicle. As objects get further away from the camera their observed vertical position tends towards a horizon point (assuming they are on flat ground on the earth's surface). Figure 3.7 shows the relationship, which was calculated by considering a point horizontally centred in front of the camera which moves from 0m to 20m away and perpendicular to the rear of the vehicle. The graph shows that the expected position of the object asymptotically approaches a value in the top half of the image which incidentally is at 147 pixels. Analysis of this graph shows that the greatest spread of vertical positions can be obtained by placing targets between 1m and 10m. However it is impractical to make targets a few mm across as needed (see Figure 3.6), thus the targets with the sizes determined by Equation 3.7 are placed at distances of around 7m (5 targets), 17m (5 targets) and 26m (1 target). The vertical spread and expected height of the four scan layers were also calculated for these distances in order to verify the targets will be visible to only 1 of the 4 layers.

The initial placement of the targets relative to each other is not of great importance since the vehicle can be moved about readily to observe the same target arrangement from several positions. Doing this effectively produces further sets of independent observations of the 11 targets which can be used for validation or further samples for calibration.

Pre-processing of the laser scanner data

Some difficulty was experienced in obtaining accurate measurements from the laser scanner due to noise from the disk scattering and stick reflection especially in the case of targets 1, 3, 4 and 5 in Figure 3.5. This problem was solved by merging 8 frames of

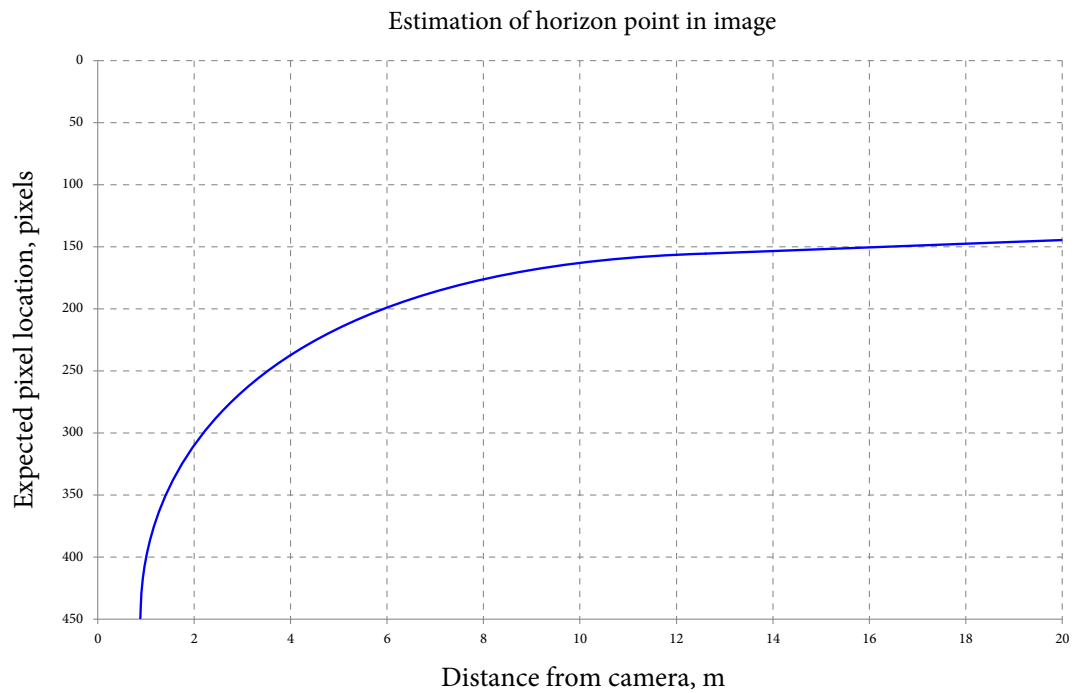


Figure 3.7: Graph to show the expected vertical position in the camera image of a target at a given distance.

laser scanner data together (consecutive frames representing about half a second with the vehicle stationary during this period) rather than looking at individual frames. Firstly, the clusters of points that are scattered around the real location of the target are grouped together using a distance threshold of 1.5m. This threshold was chosen as it is less than half the average distance (4m) between the targets and avoids points being assigned to the wrong group. Secondly, each cluster of points containing all points relating to the target in the 8 consecutive frames is filtered to remove all measurements from layer 0.

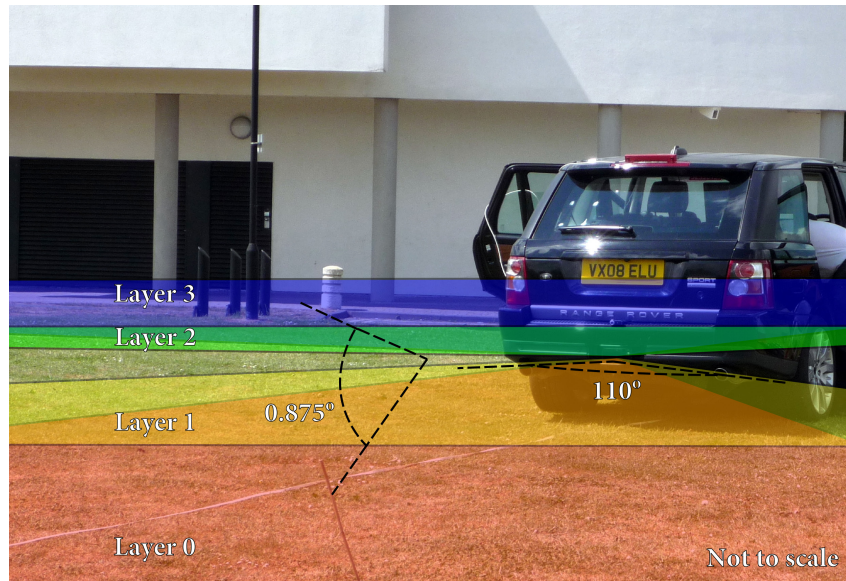


Figure 3.8: ibeo LUX scan layer diagram showing the 4 layers spread 0.875° apart giving a total vertical spread of 3.5° with a horizontal scan arc of 110° .

The laser scanner has 4 layers with a vertical spread of 3.5° centred about a plane parallel to the road. The layers are numbered 0–3 in height order where layers 0 and 1 are pointing at the road and layers 2, 3 are pointing above the road as shown in [Figure 3.8](#). Thirdly, the remaining points are sorted by their reported reflectivity assuming that measurements reflecting from white disk will have greater reflectivity than the stick or grass. The top 50% of these points are kept and the rest discarded. The final position of the target is taken to be the average of these remaining points using their associated reflectivity as a weighting factor. This processing resulted in the more accurate target positions shown as blue circles in [Figure 3.5](#).

Determination of the initial estimation of the parameter set

[Table 3.2](#) shows the initial estimates used to begin the optimisation. The camera position was measured with a tape measure approximately 1.7m above the laser scanner, 10cm

to the left and 20cm behind as illustrated in [Figure 3.4](#). The design of the mount used to attach the camera to the vehicle specified a vertical tilt of 21° about the x-axis so this figure that was used for the initial estimate. The image centre (320, 240) was used as the initial estimation of the optical centre coordinates in the image plane. The scale factor f was taken from the camera manufacturer's specifications. The lens distortion coefficient was estimated by using the distortion correction function in a commercial software package which applies a first order inverse model to the image. The image was corrected by eye as only an estimated value is required. The value that gave the most rectified appearance was -0.38 . The second and third order terms are expected to be near zero since the first order inverse distortion gives satisfactory results indicating there is a negligible amount of higher order distortion.

Table 3.2: Initial values for calibration parameters.

PARAMETER	INITIAL ESTIMATE
$C_{(x,y,z)}$	0m, 1.7m, 0m
$\theta_{(x,y,z)}$	$-21^\circ, 0^\circ, 0^\circ$
u_0, v_0	320px, 240px
f	266px
k_1, k_2, k_3	$-0.38, 0, 0$

3.6 RESULTS AND DISCUSSION

3.6.1 Calibration results

The calibration computation was performed by using the sample data acquired from the 11 targets as shown in [Figure 3.5](#). [Listing 3.1](#) shows the output from the optimisation program and gives the calibration results. The optimisation was allowed to run for 2

Listing 3.1: Optimisation report associated with the results in [Figure 3.9](#)

```

1 Pass 1:
2 ...(text trimmed)
3 Avg. alignment Error: 0.7583px
4
5 Pass 2:
6 Elapsed time is 0.846532 seconds using 11 points.
7 Final sse Error: 2.7818, Iterations: 2083, Function Evaluations: 2920
8 Avg. alignment Error: 0.52096px
9
10 Calibration Result:
11 CamPos: -0.086345 m, 1.630175 m, -0.215722 m
12 CamAng: -24.3307deg, -2.6687deg, -0.05262deg
13 u_0,v_0: 320.0 px, 240.0 px
14 f: 276.135177 px/m
15 Distortion: k1=-0.3658, k2=0.0741, k3=-0.0075

```

passes i.e. it was run on the initial estimates and then the results from the first run were used as initial estimates for a second run which improved the average alignment error from 0.8px to 0.5px. No more than two runs are needed since a third pass using the results as the estimate yielded no further improvement.

A comparison between the calibration results and the initial estimation in [Table 3.2](#) reveals the resulting x-axis rotation of 24° is not a perfect match, but it may be that this is closer to the true angle and that the camera was not fitted at exactly 21° as specified by the initial design specification. This is expected since it is not uncommon for fittings built by hand to deviate slightly from specifications as they lack the tight tolerances of mass produced items. As for the image centre that is clearly correct and shows that there is no misalignment of the optic centre, the scale factor f is within 4% manufacturer stated figure which is reasonable and can again be attributed to manufacturing variation. The lens distortion remained close to the initial estimate of -0.38 (in the first order) which is a close match when considering the fact that the higher order term was used for fine adjustment which was not practical when correcting by hand.

3.6.2 Validation of results using initial conditions

Figure 3.9 shows the image captured from the camera in the real setup with white markers indicating the observed target locations and coloured rings indicating the projected points. The targets themselves are too small to be seen clearly, so dots are placed at their observed locations for improved visibility. The colour of the rings corresponds to the colour bar on the side representing the amount of alignment error for each point. The rings are the projected laser scan points after the transformations using the optimised calibration parameters. After calibration the average alignment error is 0.52096. The 3 worst errors, indicated by the 3 red circles in the centre of the figure, have the values of 0.9844, 0.9209 and 0.6665. It is important to note that all target data points observed by the camera were stored as integers i.e. the pixel index in the image.

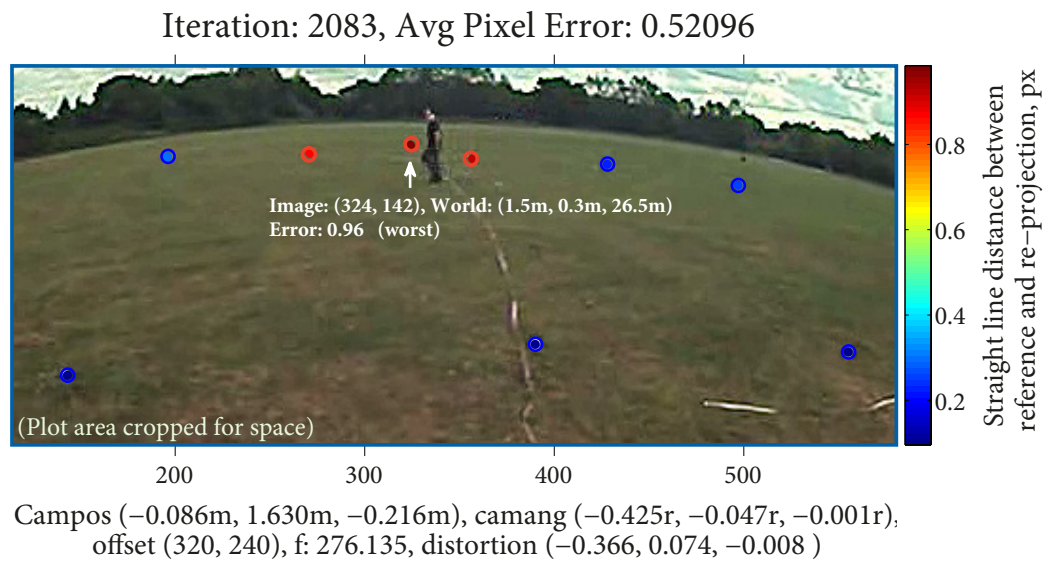


Figure 3.9: Results of calibration with lens distortion model. Average error 0.52px, largest error 0.98px.

However, projected laser scanner points are the result of a calculation and therefore are floating point numbers which are not rounded, or truncated, before being compared to the image observations for error calculation. This means there will always be some rounding error, as the projected points are unlikely to fall exactly on an integer pixel position even in the case of a perfect alignment. Due to this inherent error it is impractical to expect a final error significantly smaller than 2px.

Figure 3.10 provides a visual demonstration of the calibration accuracy. Using the same vehicle and equipment setup shown in Figure 3.4 images and laser scans were captured simultaneously in a typical urban environment and the laser data has been projected onto the image using the results obtained in the experiment. The colour of the points indicates the distance of each object from the vehicle, it can be seen that the laser points accurately coincide with the car in the distance, the pole on the right and the bench just behind it.

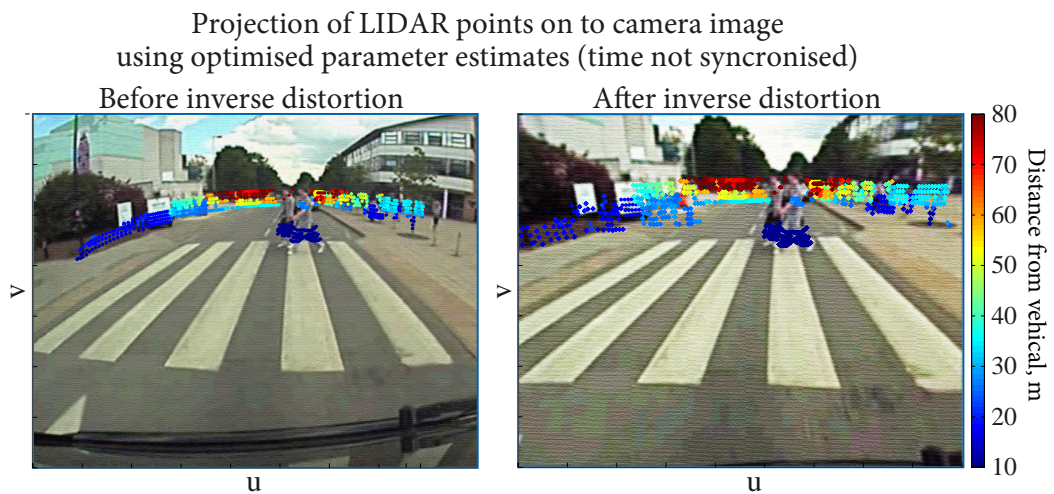


Figure 3.10: Demonstration of calibrated sensor fusion data showing the accuracy of laser scanner projection onto the image.

3.6.3 *Validation using independent samples*

A more meaningful way of validating the results is to apply the calibration parameters to a totally independent scene i.e. with samples that were not used in the calibration process. During these experiments, the vehicle was moved closer to the targets and a new independent set of samples was taken from the two sensors. The laser scanner samples and image samples are now different to the ones used to determine the optimised calibration parameters but if the calibration was successful and accurate it should align these unseen samples equally well. [Table 3.3](#) show a complete set of sample data from the new location along with the projected image points and the associated error. The rows in the table are sorted in the order of the error, worst error at the top. The average error is 1.0286px with a worst case of 1.5059px. This error is slightly larger than the first set of data, as would be expected since it was not used as a fitting target. These small errors occur in the cases where a target appears to occupy more than a single pixel, the choice of which co-ordinate to use for the sample will make the difference between zero error and 1px error. In addition, the first row of this table illustrates the problem of the integer and non-integer representations of image location. The vertical position of the sample in the first row is 184 and the projection at 183.5. If the projected points were rounded to the nearest integer, it would eliminate a portion of the error.

3.7 FUSION AND MERGING OF SENSOR DATA

This section will briefly describe how, after calibrating the sensors using the method above, the distance data from the two devices is fused together into a single, enhanced source of depth information. To reiterate, the purpose of using the two sensors together is to eliminate the weaknesses inherent to each. The method is not described in detail as it does not contain novel ideas, it simply serves to improve the depth information to level where physical properties of surfaces can be measured for classification.

Table 3.3: Independent samples and resulting projection for calibration validation.

TARGET	L			N			O		ERROR (px)
	LASER SCANNER SAMPLES	PROJECTED IMAGE	PNTS	IMAGE SAMPLES	DISTANCE FROM				
ID	x (m)	y (m)	z (m)	u (px)	v (px)	u (px)	v (px)	O to N	
2	-1.58	0.29	6.02	243.4	183.5	242	184	1.5059	
7	-1.70	0.14	18.28	283.3	147.6	283	149	1.4236	
11	1.50	-0.30	26.48	323.5	144.8	323	146	1.2917	
10	9.66	0.13	16.07	455.2	156.9	454	157	1.2218	
3	0.66	0.32	6.03	339.4	180.0	340	179	1.1627	
4	2.81	0.30	6.46	418.7	179.1	418	180	1.1431	
5	6.50	0.39	5.90	524.9	191.0	525	192	1.0482	
9	5.89	0.14	17.11	398.6	150.4	398	151	0.824	
8	2.46	0.11	18.04	345.3	147.8	346	148	0.7404	
1	-6.03	0.37	6.03	108.9	195.5	109	196	0.4954	
6	-6.00	0.10	19.02	225.1	150.5	225	150	0.4577	
MEAN ERROR (px)									1.0286
ϵ (SUM SQUARE ERROR)									12.686

Stereo alone provides dense, good quality depth information for nearby surfaces but suffers from strong noise at greater distances, more importantly it also fails to produce any kind of depth information on surfaces lacking visual texture either naturally or caused by over exposure of the image. The problem of missing depth information on overexposed surfaces is of particular trouble in this research as the road surface (which is the most important area to classify) often suffers from glare when in direct sunlight. By using the [LIDAR](#) aimed at the road some distance ahead, sparse but accurate distance information can be used to confirm, or correct, any gaps in the depth map produced by the stereo. The [LIDAR](#) does not generally suffer from ambient lighting conditions and has significantly lower error margins at distances up to 200m. Since the depth information from the [LIDAR](#) is sparse, i.e. there is not depth information for each pixel in the image, linear interpolation is used to fill in the gaps.

The method used to do this is a simplified version of the work done by Badino et al. [13]. In that paper, the authors describe using interpolated [LIDAR](#) data to estimate upper and lower bounds for the depth of each pixel in the stereo depth map. The method requires actually modifying the disparity algorithm to take into account this information but since in this case the disparity processing is performed by the proprietary Point Grey software, “Triclops”, it is not possible to directly modify its behaviour. Instead the interpolated [LIDAR](#) data was used to patch or replace any region of the computed disparity map where [LIDAR](#) data was present but stereo data was missing or significantly different to what the [LIDAR](#) measured. In areas where no [LIDAR](#) data was present the stereo data was left untouched.

3.7.1 *Final calibration*

Before interpolating the [LIDAR](#) data it needs to be transformed via perspective projection to get the distance measurements onto the same plane as the stereo data. [Table 3.4](#) shows

the result of using the proposed calibration method on the actual data collection rig. Figure 3.11 shows a photograph of the Range Rover fitted with the two devices with the offsets marked.

Table 3.4: Final calibration parameters used during the acquisition of all datasets.

PARAMETER	FINAL VALUE
$C_{(x,y,z)}$	0m, 0.99m, -1.04m
$\theta_{(x,y,z)}$	$-0.006^\circ, 0.07^\circ, 0.005^\circ$
u_0, v_0	241px, 166px
f	793.4px
k_1, k_2, k_3	0, 0, 0

The LIDAR was fitted such that the scan layers intersect the road at approximately 30m ahead, the distance at which stereo depth information begins to show unacceptable levels of noise.

3.7.2 LIDAR interpolation

In the paper by Badino *et al.* [13] a full three dimensional dome scanner was used i.e. it scans over a wide arc in both vertical and horizontal directions. This gives LIDAR coverage over the full scene observed by the stereo camera setup. This is not the case with the hardware used in this research, instead the ibeo LUX LIDAR has a limited vertical area of only 3.5° [64]. This means only a narrow horizontal strip of the depth map can be scanned and verified by the LIDAR.

Figure 3.12 shows how the linear vertical and horizontal interpolation is done with this dome scanner, the same approach applies to the LUX.

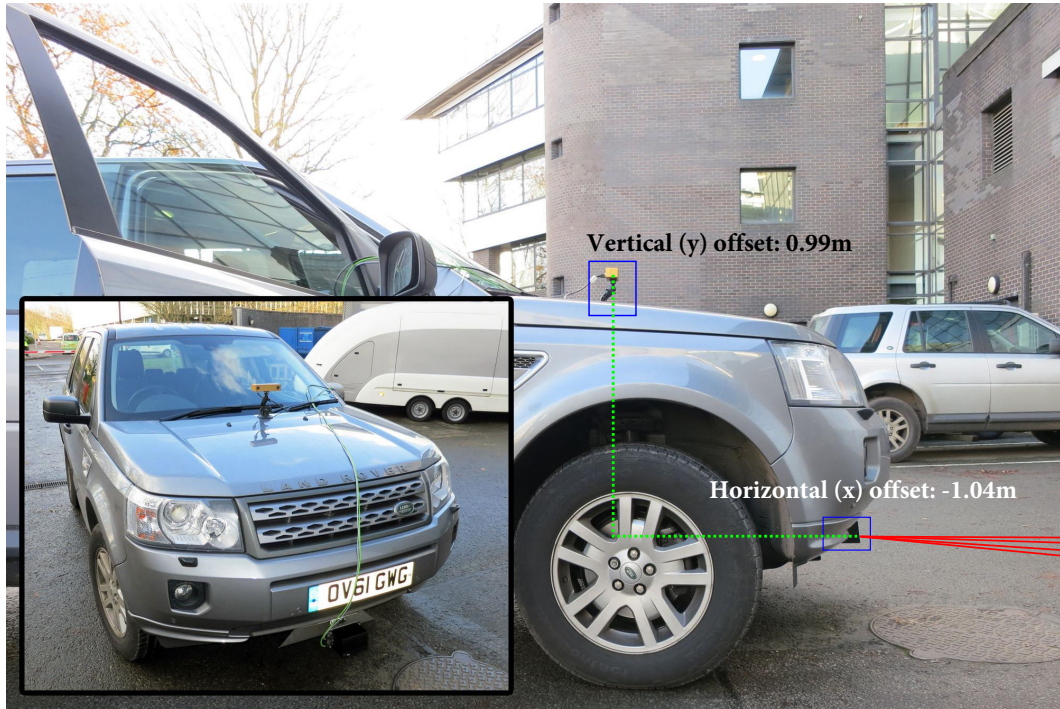
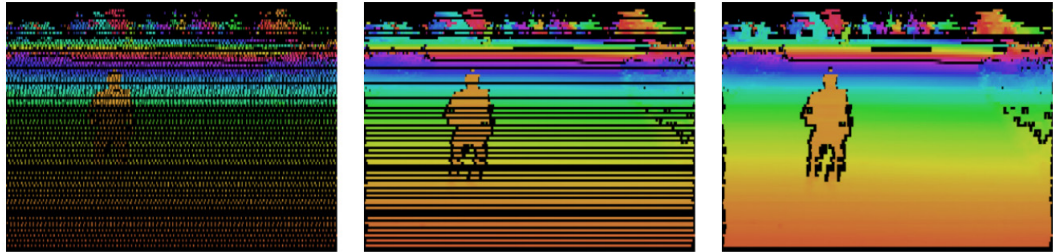


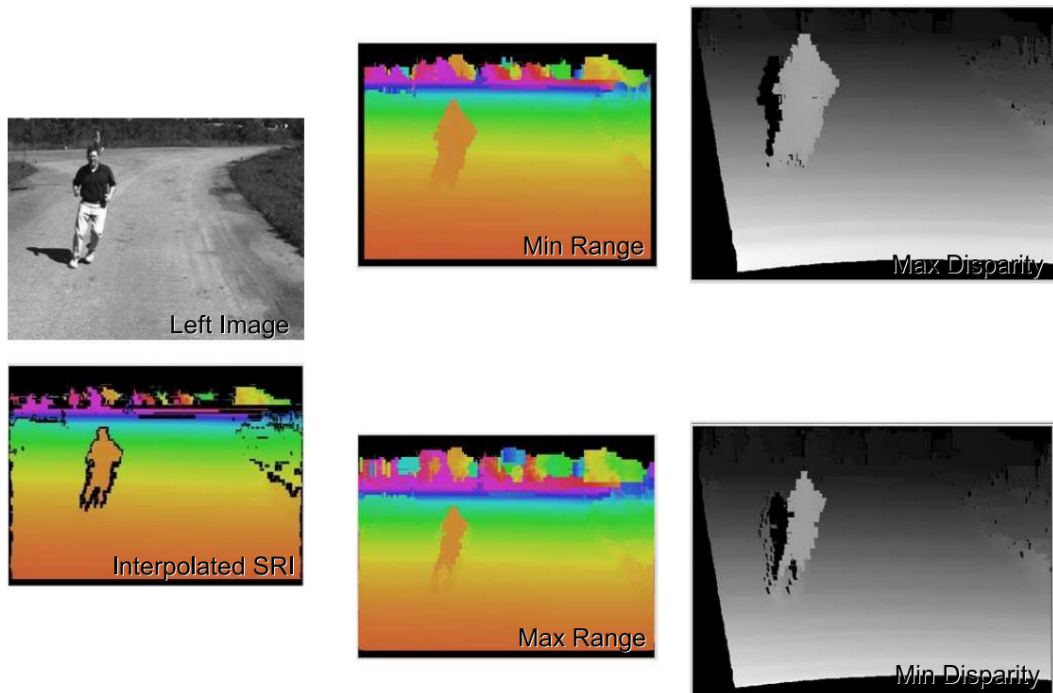
Figure 3.11: Sensor mounting positions on data collection vehicle showing camera to [LIDAR](#) offset in x and y directions as determined by optimisation procedure, other calibration parameters can be seen in [Table 3.4](#).

In the paper the interpolation is done in Spherical Range Image ([SRI](#)) space where each point in the disparity map is represented by the horizontal angle, vertical angle and range from the [LIDAR](#) (θ, ϕ, r) instead of image row, column and disparity. Gaps between [LIDAR](#) points are interpolated linearly and any large gaps, or depth discontinuities, are excluded thus preserving the black occluded zone around the pedestrian. With so many data points from the dome scanner it is possible to produce a dense interpolated depth image. Using this, and the expected error on the [LIDAR](#) at the range of each point, an upper and lower bound can be computed on the expected distance of that point. [Figure 3.12 \(b\)](#) visualises the difference between these limits when the bounds are converted back into pixel disparity units produced by the stereo matching algorithm.

Using the ibeo LUX, the approach had to be modified since the amount of data available is significantly reduced and once the [LIDAR](#) scans have been projected onto the image



(a) Illustration of horizontal and vertical LIDAR interpolation using a full 3D dome scan LIDAR, colour encodes the distance. From left to right, raw LIDAR data, after horizontal interpolation, after both horizontal and vertical interpolation.



(b) Using the performance characteristics of the LIDAR i.e. the known distance resolution and accuracy, maximum and minimum bounds on the error can be computed and used to override the stereo disparity in cases where it does not fall in this interval.

Figure 3.12: Illustration of horizontal and vertical LIDAR interpolation using a full 3D dome scan LIDAR, these figures were created by Badino et al. [13].

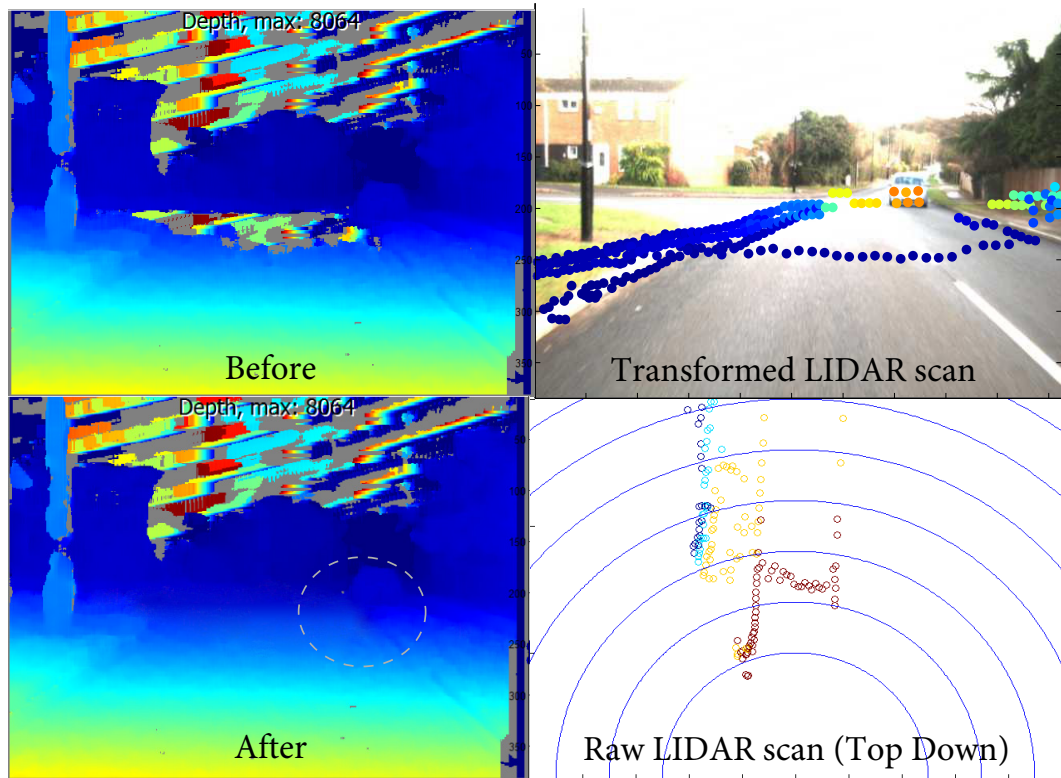


Figure 3.13: Improved stereo after integrating LIDAR depth information. Top left shows raw disparity image, top right shows projected LIDAR scan using calibrated transform, lower left shows the depth image with LIDAR interpolated fill for missing centre area, lower right shows raw LIDAR scan on $X-Z$ axis, radius grid spacing is 2m. The grey dashed circle shows a discrepancy between the interpolated and expected road contour.

plane they do not tend to form orderly rows of data points, instead they are scattered across the image in the vertical direction due to the low vertical angular resolution.

The top right section of Figure 3.13 shows how sparse the data from the LUX is once projected. In this situation it is not practical to exclude large gaps during the horizontal interpolation stage, otherwise the result would be a single interpolated line through the middle, with no reference points available to expand the interpolation vertically. Instead horizontal interpolation is done whenever there are two or more points on the same row of the image which allows a large region of road to be depth estimated. This is only applied across large gaps when there is not also a large discontinuity in depth to prevent filling regions with false gradients. Since roads are generally wide flat regions this approximation

tends to produce good results. [Figure 3.13](#) shows a particularly good example of the benefit provided by the fusion, in this frame the image has been over exposed causing the road to reach white saturation and, as a result, the disparity algorithm has failed to produce meaningful depth information in the region just behind the car. Fortunately the [LIDAR](#) has scanned the road just behind, ahead and each side of the region allowing a fully interpolated representation to be created. The [LIDAR](#) interpolated depth image covers only a small patch of the image but, in this case, it has replaced the missing and noisy (random colour areas) regions just behind the car with a smooth gradient which by appearance looks correct.

There is a slight discrepancy inside the grey dashed circle where the interpolated [LIDAR](#) region meets the original stereo depth map which gives the impression of a dip in the road which is not actually present. Since there are no [LIDAR](#) reference points at this particular vertical location in the image the interpolated path does not perfectly match up with the true curvature of the road.

3.8 THEORETICAL STEREO NOISE PERFORMANCE

The theoretical error in stereo measurements is proportional to the square of the distance measured as shown in [Equation 3.9](#). Using complete, dense disparity images a short comparison was conducted to observe the extent of error between distances measured by the LIDAR and the same distance measured by the stereo to get an idea of the calibration accuracy when applied to fusion.

$$\begin{aligned} f_s &= f_r \\ \epsilon &= \frac{z^2 c}{f_s b} \end{aligned} \tag{3.9}$$

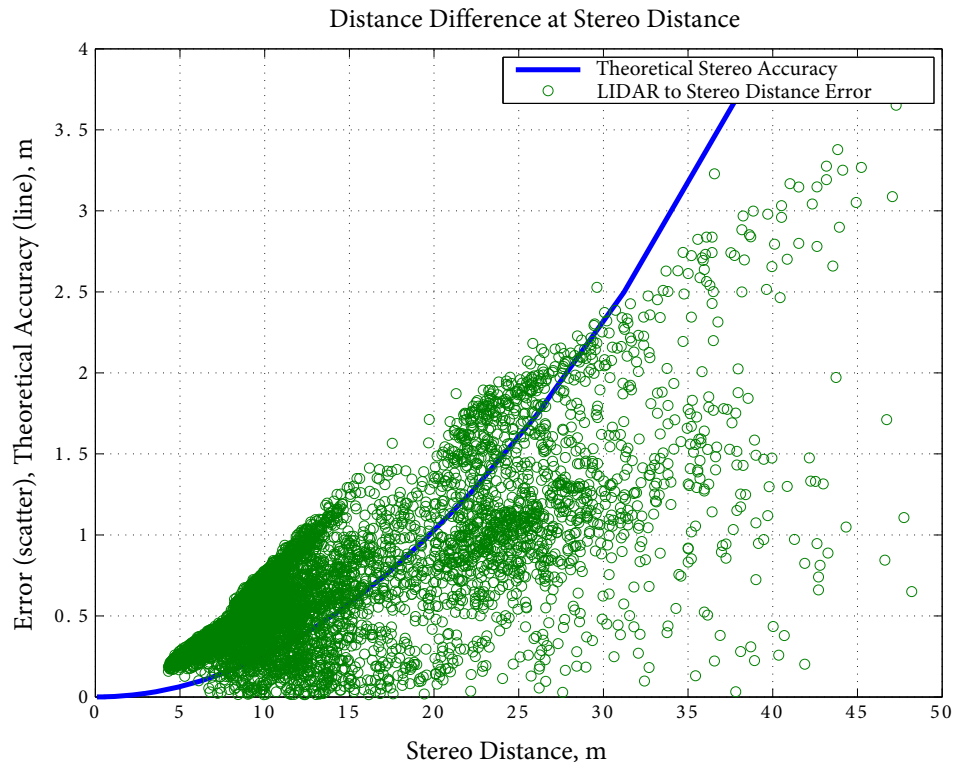
where (including physical values from Bumblebee technical manual where appropriate):

- b baseline distance, value 0.12 Inc. [85]
- f normalised focal length, value 1.27 [85]
- r horizontal resolution of image, value 512 (images scaled by half)
- c root-mean-squared accuracy of correlation in pixels, value 0.2 [85]
- f_s focal length of stereo in pixels
- z distance of a point in question, m
- ε distance error (accuracy), m

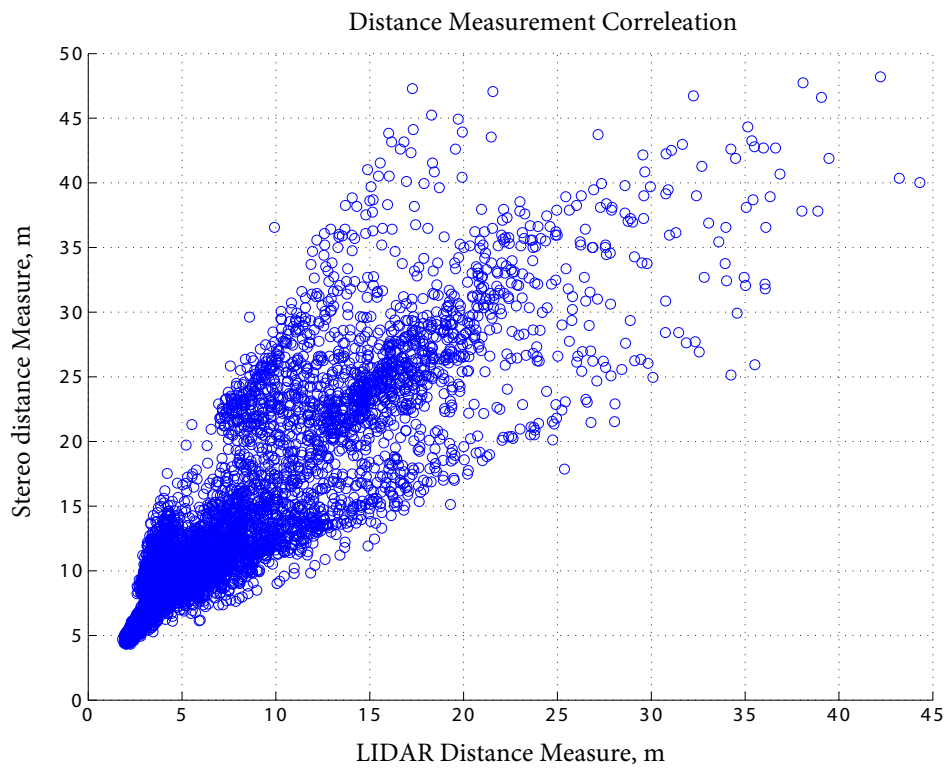
Without the necessary tools to create a ground truth comparison of the range data from each sensor this method can give some idea of the performance. Taking 10 frames from the “Berkswell manual” dataset (typical rural roads, see [Section 4.3.4](#)) where the disparity coverage was sufficiently dense to cover the same image points as the projected [LIDAR](#) data the distance measured by each device was recorded. To do this the disparity image was converted to an [SRI](#) then the range of each depth-image point, covered by a [LIDAR](#) point, was extracted for comparison to the reported range a each [LIDAR](#) point.

[Figure 3.14](#) shows a scatter of distance differences between the two distance measures. There are many factors which would lead to discrepancy between the two measurements in addition to assumed measurement noise such as alignment shift in the sensors and more significantly temporal misalignments i.e. the data from each sensor being captured several milliseconds apart. While [Figure 3.14](#) (a) does show some similarity to the curve produced using [Equation 3.9](#) the true pattern of error against distance looks to be linearly bounded. This is because the error measured is not true error but the error between the two devices, while the stereo error increases with the square of the distance the [LIDAR](#) error will be more uniform.

[Figure 3.14](#) (b) shows the correlation of the measurements which, if each sensor was perfectly accurate, would be a straight $x = y$ graph. The distribution of points clearly



(a) Comparison of theoretical stereo distance error computed using Equation 3.9 with actual distance error compared against LIDAR measurements.



(b) Correlation of distance measurements between the two devices.

Figure 3.14: Selection of graphs showing a comparison between the theoretical and observed errors in stereo distance data when compared to LIDAR measurements.

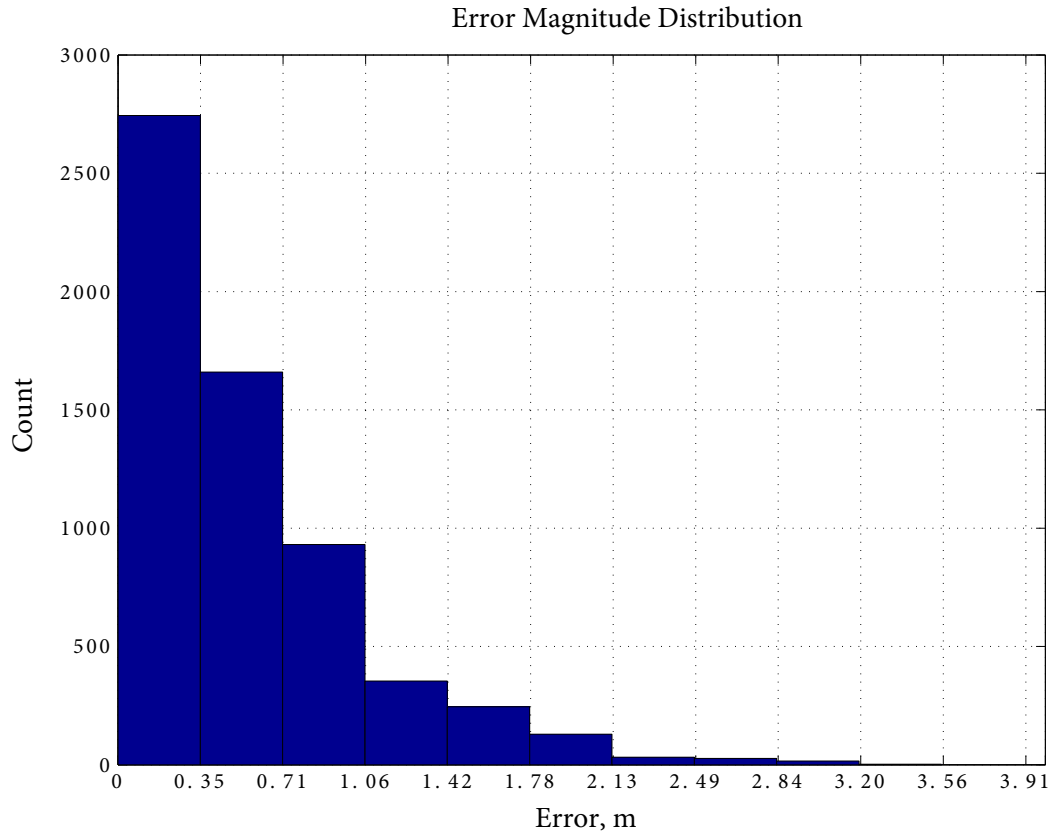


Figure 3.15: Histogram of error magnitude shown in [Figure 3.14](#) (a) and (b).

shows the errors growing at the longer ranges. According to the ibeo LUX handbook [64] the [LIDAR](#) has a range independent accuracy of 10cm therefore it is reasonable to assume the error is mostly attributed to the stereo camera.

The most important factor to consider here is the average severity of the errors as shown in [Figure 3.15](#). Since most of the errors are between 0cm and 70cm and at least 10cm should be expected at any range due to [LIDAR](#) uncertainty the accuracy is sufficient for establishing basic geometric information. At longer ranges the errors are within theoretical limits and the quantity of relatively large errors shown on the histogram is small compared to the majority, indicating they are simply noise. Together this information indicates that the calibration is holding well across the frames examined.

4

DATA COLLECTION AND SET DESCRIPTION

CONTENTS

4.1	Data collection hardware	84
4.1.1	The ibeo LUX LIDAR	84
4.1.2	Videre STOC stereo on chip system	85
4.1.3	Point Grey Research bumblebee2	87
4.1.4	Computer hardware	89
4.2	Notes on data collection	89
4.3	Data sets	95
4.3.1	The “A46” set	95
4.3.2	The “RX Residential” set	96
4.3.3	The “Berkswell Auto” set	97
4.3.4	The “Berkswell Manual” set	98
4.3.5	Sample images from the sets	99

When choosing locations for data collection the aim was to try and get a good representative sample of several driving environments. The most important ones being motorway driving, residential areas, sparse urban areas and rural settings. For each of these settings, data was recorded during a drive lasting approximately 20mins to make sure that a large quantity of high quality frames would be available later for classifier training. First, a brief description of the hardware used in this data collection task.



Figure 4.1: Land Rover used for data collection.

4.1 DATA COLLECTION HARDWARE

The vehicle used for data collection was a Range Rover shown in [Figure 4.1](#), the camera was attached to the bonnet using suction cups and the Light Detection And Ranging ([LIDAR](#)) was attached below the front bumper on a custom built mount.

4.1.1 *The ibeo LUX LIDAR*

After researching a number of available [LIDAR](#) devices it was found that most models are not suitable for driving applications due to insufficient range, or insufficient angular resolution. The ibeo LUX is designed specifically for this application and possesses, by far, the best set of features compared to currently available devices, see [Figure 4.2](#).

The purpose of the [LIDAR](#) in this application is to assist and enhance the stereo camera system via sensor fusion. Since the stereo is already accurate up to distances of 30m there is no benefit in fitting a [LIDAR](#) with a range less than this. What is needed is a [LIDAR](#) which

can compensate for any weaknesses in the stereo system. The most important surface in the scene which needs accurate depth information is the road, unfortunately it also the surface most commonly missing in the stereo depth information due to its smooth visual texture. Therefore it would be ideal to have a [LIDAR](#) which scans only the road surface and ensures that depth information is always available regardless of lighting conditions.

The ibeo LUX fits this requirement well, it has a range of 200m, an angular resolution of 0.25° and a horizontal Field Of View ([FOV](#)) of 110° . It is fitted on the vehicle such that it sits parallel to the road. Because it has four vertical layers spread over 3.5° , at least one of the layers will be scanning the road even when the vehicle tilts during acceleration and breaking. It can be set to scan at rate of up to 50Hz however a rate of 12.5Hz was chosen for this data collection. Finally the device connects to the on board computer via Ethernet and has easy to use software which streams the data to disk.

4.1.2 *Videre STOC stereo on chip system*

The Stereo On Chip ([STOC](#)) is an experimental new device shown in [Figure 4.3](#). It functions as a complete vision processor, combining stereo cameras and an embedded processor to directly produce 3D stereo information. The processor is a specialised Field-Programmable Gate Array ([FPGA](#)) which implements a dense disparity map generation algorithm. Because the embedded processor has been designed for just this task, it is more capable than general purpose PC microprocessors. Even modern computers will struggle to perform the disparity calculation in real-time (30 fps) depending on the algorithm. The result is an output stream of disparity information, at a resolution of 640 by 480, at 30 frames per second. The stereo cameras are 640 by 480 pixel, progressive scan Complementary Metal-Oxide-Semiconductor ([CMOS](#)) imagers mounted in a rigid body. They have a global shutter – all pixels are exposed at exactly the same instant. This makes the [STOC](#) suitable for environments with fast movement, such as on an outdoor

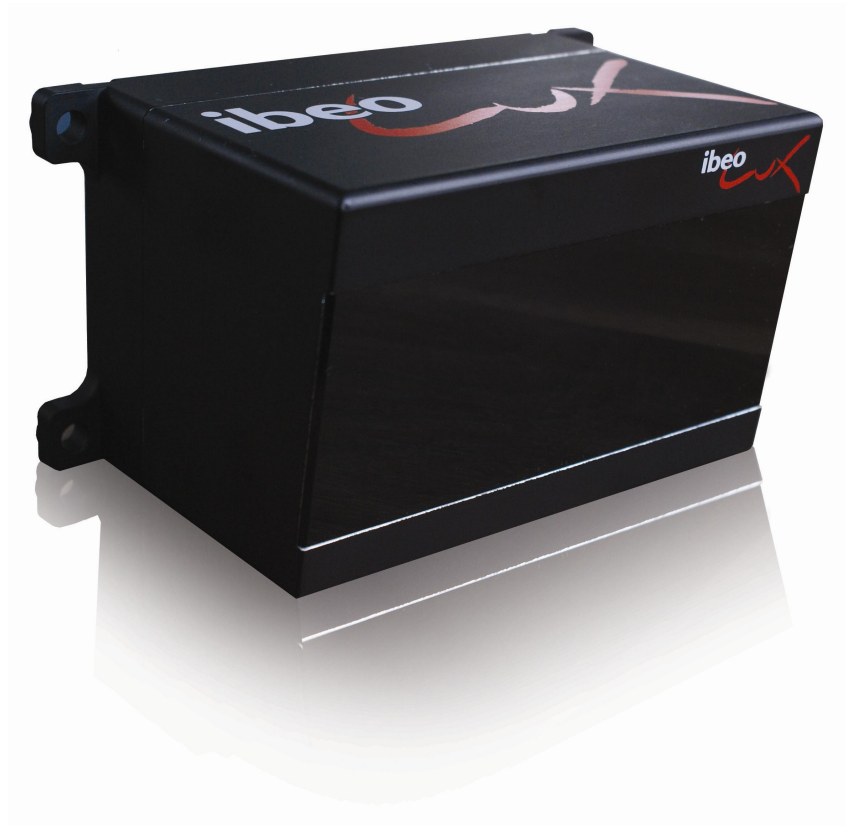


Figure 4.2: ibeo LUX four layer [LIDAR](#).

vehicle. Another important feature of the [STOC](#) is that it is fully programmable: the user can set exposure, gain, decimation and even adjust the calibration.

After using this device for several months it was concluded that it was unsuitable for the project for a number of reasons. The most critical issue is that the camera can only provide 2 image streams simultaneously, so for example if the left image and depth data are being transmitted, then the right image can never be recorded. Also intermediate images, such as rectified images and colour versions of the images, are not available at the same time, only 2 may be chosen for further analysis on computer. Secondly it was determined that a resolution of 640 by 480 was actually insufficient to get even reasonable depth accuracy at a range beyond 15m. Another problem was that there was no way to



Figure 4.3: Videre [STOC](#) stereo on chip system.

externally sync, or trigger, the camera which means syncing the samples with the [LIDAR](#) was impossible.

Finally the lack of support for modern operating systems made it difficult to test and run alongside other data capture software such as the [LIDAR](#) recorder. Since the company producing these is no longer trading it is safe to assume support will never be added for Windows 7.

4.1.3 *Point Grey Research bumblebee2*

The Point Grey Research Bumblebee2 shown in [Figure 4.4](#) is a stereo vision device which focuses on image quality. Instead of attempting to be an all-in-one device it only captures



Figure 4.4: Point Grey Research Bumblebee2.

and transmits image data. The camera comes with a software package called the “Triclops API” which uses a proprietary stereo algorithm to produce disparity maps from the pair of images provided by the camera. The software is particularly useful because it allows easy interfacing with Matlab (the primary programming tool used in this thesis) and allows precise control over how images are processed. For example the user can choose to have the images rectified, colour converted and disparity mapped using only a few lines of code. In addition it can also use its internal calibration data to convert any disparity map into a 3D point-cloud taking much of the work out of low level data manipulation.

The model used in this research is a BB2-08S2C-60 with a 43° FOV and a pair of 1024 by 768 imaging sensors. It connects using Firewire and uses standard video transmission protocols meaning it can be used with any generic video capture software which supports Firewire cameras – another benefit.

4.1.4 *Computer hardware*

The data collection computer needs to be able to receive, compress and store a continuous stream of video data from the camera and a continuous stream of 3D point data from the LIDAR simultaneously. Depending on the compression method used this task requires considerable processing power. In addition, the amount of data to be stored on a per second basis is very large and, in general, laptop HDD's do not perform as well as their desktop counterparts, due to durability considerations.

For these reasons the computer chosen to record the data was an HP EliteBook 8540w running Windows 7. This laptop represents the top-end of the performance spectrum within the range of currently available laptops. The laptop is fitted with an Intel i7-620M 2.66GHz quad-core processor, 8GB of memory, an on-board Firewire socket for the camera and a Gigabit network controller for the LIDAR. The standard HDD which ships with the laptop is not sufficiently fast to record data at the rates required, so this was replaced with a mid-range SSD drive. Although the quad-core CPU performs well in multi-tasking scenarios, most video compression codecs do not take advantage of multiple threads. For this reason the CPU does not cope well with the more computationally intensive video codecs. For those looking to perform similar data collection tasks it is recommended to choose a dual-core CPU with a higher clock speed. Despite this problem the upgraded laptop generally fits the data capture requirements well, having all necessary input ports to connect the sensors.

4.2 NOTES ON DATA COLLECTION

Before settling on a final protocol for recording the data a number of issues had to be considered and dealt with appropriately. This subsection briefly describes some of the trade-offs that were made.

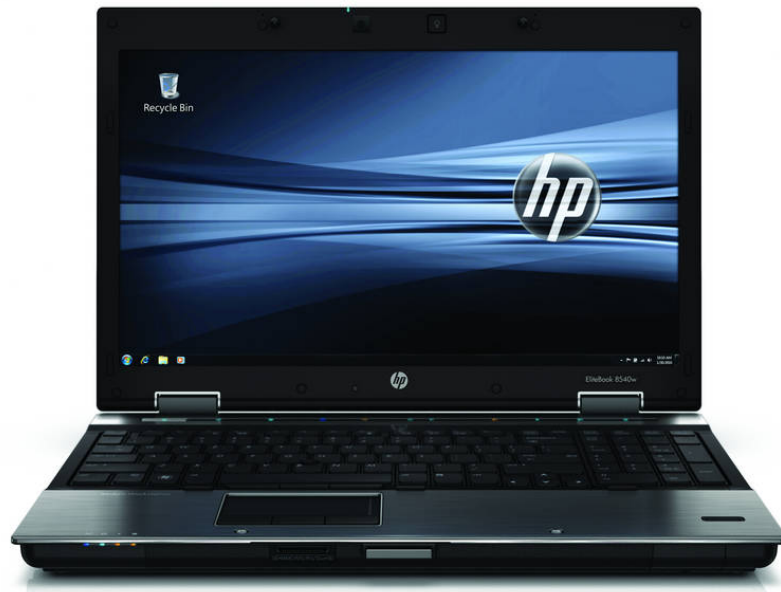


Figure 4.5: HP EliteBook 8540w.

AUTOMATIC GAIN CONTROL The automatic gain control system integrated into the firmware of the bumblebee2 appears to have been designed to adjust the gain such that it minimises the quantity of saturated (high or low) pixels in the image. In general use this is desirable but, in the case of stereo vision, it severely degrades the performance of the disparity algorithm by making the image too dark in scenes where there are large variations in light intensity. This is the case for most images since the top part of the frame will be of the sky, which is always bright compared to the ground surface. In order to prevent the sky from being over exposed the Automatic Gain Control ([AGC](#)) reduces the gain to the point where parts of the road are underexposed for example [Figure 4.14](#). The figure shows that the clouds are preserved but the road is too dark and the stereo quality is poor. Now compare this to [Figure 4.15](#) where manual gain control was used and see how the sky (which is unimportant in relation to this work) is saturated to white, the road has better definition on the surface and the stereo quality is nearly perfect. These two images

were taken at approximately the same spot on the road from the “Berkswell” sets (as marked on the map with a red circle) and on the same day less than an hour apart. For this reason manual intervention was required to adjust the gain during the recording to get the best results.

MOTION BLUR On a related note, once the manual gain control method was adopted it also allowed manual adjustment of exposure time on the camera to be carried out. Experiments showed that keeping this as low as possible, without using excessive gain, produced the best results. The reason for this can be seen in [Figure 4.12](#) where, although brightly lit, the road surface is not being adequately mapped for disparity, due to the loss of texture. This happens when moving at relatively high speeds due to the motion blur. As with any photographic technique, reducing the exposure time helps alleviate this problem.

VIDEO CODECS AND COLOUR PROCESSING Video codecs compress the image data from the camera in real-time to store the data on disk. The choice of codecs generally comes down to a trade-off between image quality, storage size and processing power to compress the data. In this application image quality is the highest priority. Since the image will be compared pixel by pixel for disparity mapping, any compression artefacts, or approximations of colour information, will severely degrade the stereo quality. This was confirmed after testing various “lossy” codecs such as MJPEG, which produce highly compressed data at the cost of image quality. In video compression a “lossy” codec is any compression method which discards some of the original data for improved compression rates. i.e. if video data is compressed then decompressed using a “lossy” codec the pixel data for each frame will not be a replica of the original. Differences between the original and the reduced quality, compressed output, are known as compression artefacts. For this reason codecs such as these were ruled out leaving the choice between a variety of lossless codecs. Lossless codecs preserve image data exactly as it was recorded

with identical reproduction, however data storage requirements are higher. First, H264 was tried as it offers better compression ratios than some others, but was later discarded because the processor on the data collection computer was not fast enough to keep up with the video stream. This resulted in many frames being dropped completely, or torn, as shown in [Figure 4.6](#). Instead HFYU was chosen since the encoding method is less CPU intensive. However, due to the large storage requirements, the HDD was not always be able to keep up with the required write rate which again led to dropped frames, this was solved by upgrading to an SSD drive capable of storing data at over 200mb/s as discussed in the first section. As can be seen in the following data set descriptions, the storage required for a 20min video is around 20GB, this is after switching the recording mode to raw Bayer (colour information encoded in grey-scale Bayer mosaic pattern [3, 2]) to reduce the image stream from 24bits per pixel to 8bits per pixel. This reduces the amount of stored data significantly but means that the Bayer encoding must be reversed offline when the images are analysed.

DECODER NOISE After deciding to use the HFYU codec, and collecting all the necessary data, it was discovered that unexpected noise patterns were being created in seemingly saturated regions of the image. For example look at the disparity map on the right of [Figure 4.15](#), the distinctive stripy diagonal noise pattern, seen in the sky, is present in all images with white saturated regions. After investigation, and consultation with Point Grey Research, it was determined that the noise had been introduced by a software bug in the HFYU decoder when converting from internally used YUV colour space back to Red, Green, Blue (RGB) colour space. Rounding errors cause regions of maximum intensity (level 255) in each colour channel to be permeated by a diagonal strips of pixels at an incorrect value of 254 as seen in [Figure 4.7](#). This leads to incorrect matches in the disparity algorithm causing tell-tale pattern. Although the problem was mostly cosmetic, knowing

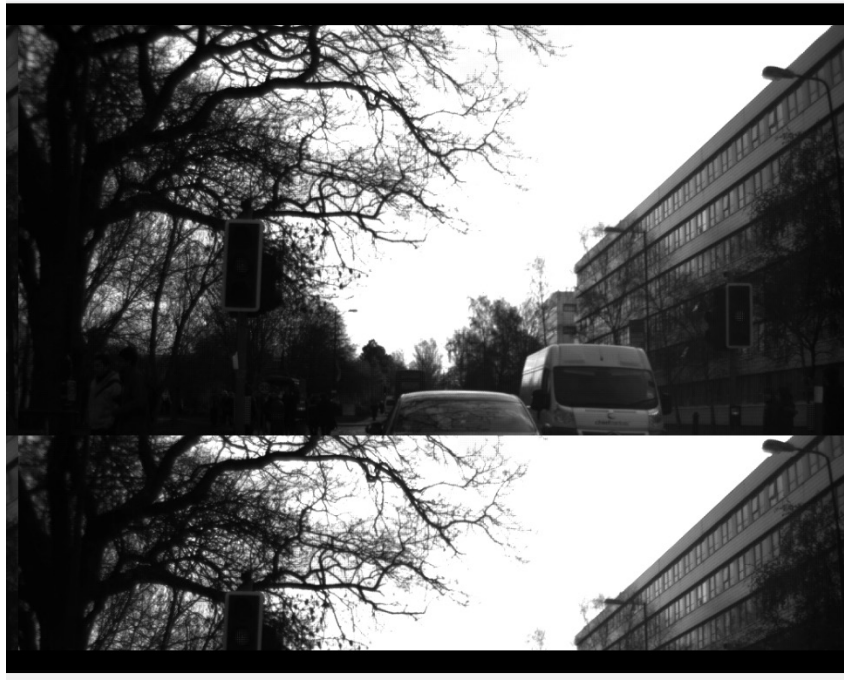


Figure 4.6: Example of image tearing due to insufficient processing power while using h264 compression codec.

exactly how the patterns were formed, made it possible to write a filter to replace pixel values of 254 surrounded by 255's with 255 thus eliminating the problem as seen in [Figure 4.13](#).

TIME SYNCHRONISATION The greatest challenge faced in getting accurately aligned stereo and [LIDAR](#) data was ensuring that each device captures information at precisely the same instant in time. This is presented some difficulty because the camera only operates at frame rates which are multiples of 7.5 and the [LIDAR](#) only operates at scan rates which are multiples of 12.5 Without the necessary hardware to trigger one sensor from the other an approximation was used. To get as close as possible the camera was set to 15fps and the [LIDAR](#) to 12.5Hz then for each video frame the nearest [LIDAR](#) frame was extracted from the dataset. Another problem is that generally video codecs do not time stamp video frames - they are just a continuous stream of frames. This means that if a frame was dropped there is no



Figure 4.7: Example of decoder noise, contrast greatly enhanced.

way to tell that all subsequent frames will be displaced in time. This was resolved by manually identifying time events in the video and [LIDAR](#) to periodically re-sync the data streams.

[Table 4.1](#) outlines the final recording parameters used for all sets, the video stream from each camera is joined side to side creating a video width

Table 4.1: Video compression settings for data recording.

ATTRIBUTE	VALUE
LIDAR rate	12.5Hz
Codec	HFYU Lossless
Resolution	2048 x 768
Data rate	193mb/s
Colour processing mode	Raw Bayer

4.3 DATA SETS

4.3.1 The “A46” set

Table 4.2: “A46”.

ATTRIBUTE	VALUE
Duration	27 : 04
Distance	11.1mi
Frames	19882
Frame Rate	15
Gain	Automatic
Storage Size	25.5GB

This was the initial drive to test the recording equipment and data capture performance. If the number of frames recorded is compared to the number of frames expected according to the duration and FPS it can be seen that there were many dropped frames as discussed above. The aim of this set was to record a dual carriage way with many other road users travelling at high speeds however due to patchy disparity data it was not used for training. See [Figure 4.12](#) on page 99.



Figure 4.8: Map of route taken for dataset “A46”.

4.3.2 The “RX Residential” set

Table 4.3: “RX Residential”.

ATTRIBUTE	VALUE
Duration	19 : 18
Distance	5.9mi
Frames	15582
Frame Rate	15
Gain	Manual
Storage Size	19.7GB

This dataset was an experiment using manual camera gain control to improve the lighting on the road surface compared to the inadequate automatic gain control system. The aim was to keep the exposure as short as possible to reduce motion blur while increasing gain until the road was well lit. This set (named after the manual gain operator) was taken in sparse residential areas.

See [Figure 4.13](#) on page 99.

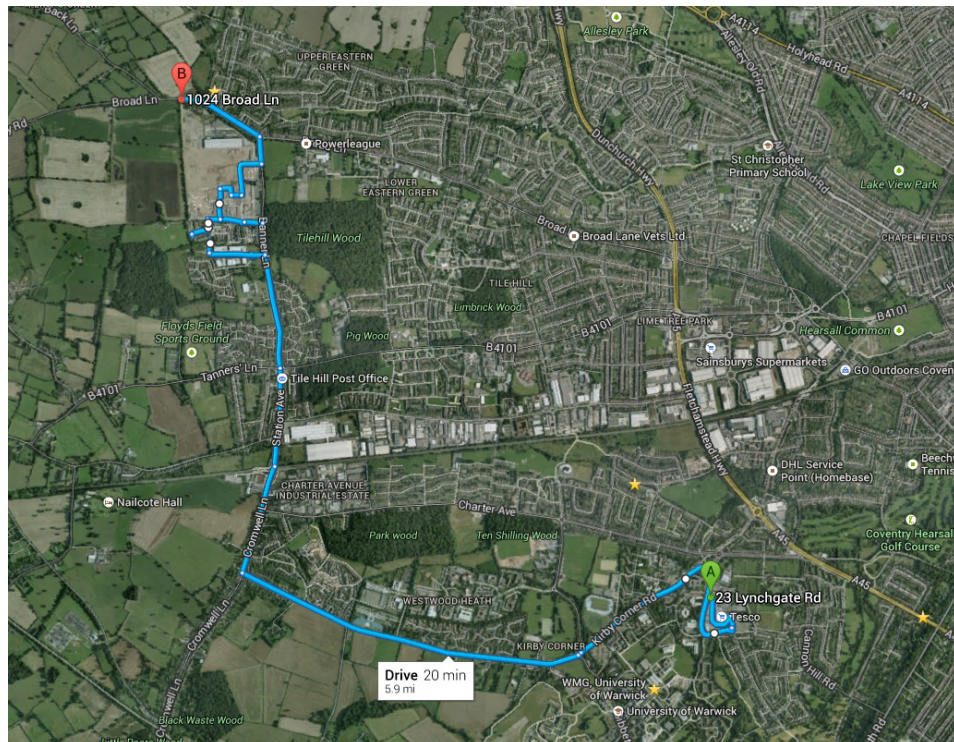


Figure 4.9: Map of route taken for dataset “RX Residential”.

4.3.3 The “Berkswell Auto” set

Table 4.4: “Berkswell Auto”.

ATTRIBUTE	VALUE
Duration	6 : 38
Distance	3.5mi
Frames	5052
Frame Rate	15
Gain	Automatic
Storage Size	6.5GB

This was intended to be a fully rural dataset but, due to the low quality of the stereo data being recorded, it was cut short. The problem was again due to poor automatic gain control on the camera. It was decided to restart the run with manual gain control but keep this data for side-by-side comparisons of the stereo quality with each method of gain control. See [Figure 4.14](#) on page 100.



Figure 4.10: Map of route taken for dataset “Berkswell Auto”.

4.3.4 The “Berkswell Manual” set

Table 4.5: “Berkswell Manual”.

ATTRIBUTE	VALUE
Duration	18 : 36
Distance	8.4mi
Frames	14974
Frame Rate	15
Gain	Manual
Storage Size	19.5GB

This is the full length rural data set, with manual gain control; the stereo data is significantly more detailed. The scenery mostly consists of narrow country lanes with plenty of vegetation, uneven road surfaces and tight turns. The route also passes through a building site where the road surface is gravel and shingle, not tarmac, which could provide an additional training class. See [Figure 4.15](#) on page 100.



Figure 4.11: Map of route taken for dataset “Berkswell Manual”.

4.3.5 Sample images from the sets

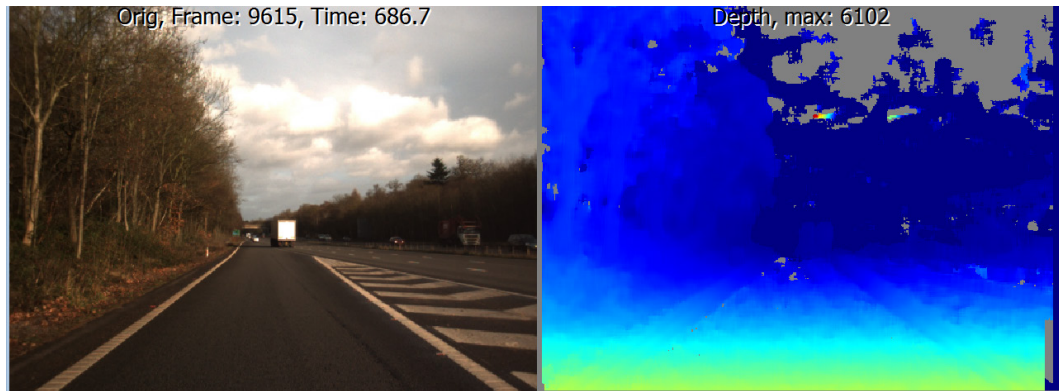


Figure 4.12: Left: image from the left camera, Right: the corresponding depth map. Sample of set “A46” showing motorway entrance ramp. Automatic exposure means that the road surface is dark and stereo information is often patchy. The [LIDAR](#) fusion helps to replace missing data with interpolated depth information.

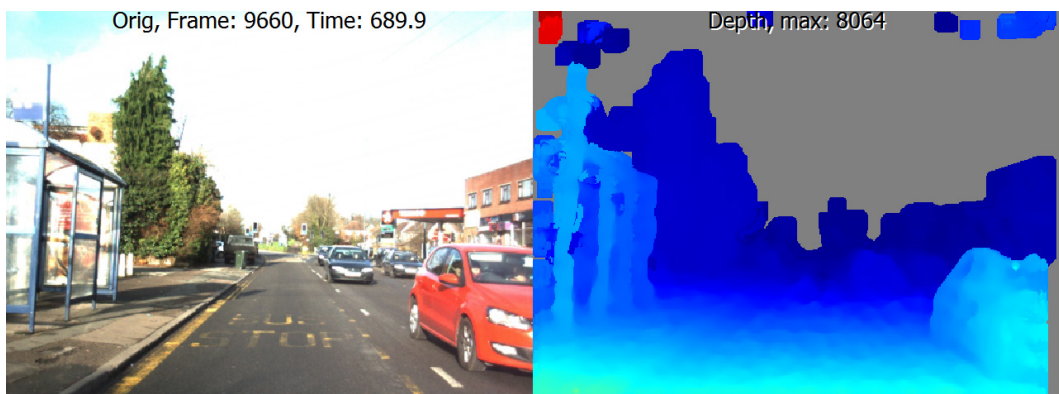


Figure 4.13: Left: image from the left camera, Right: the corresponding depth map. Sample of set “RX Residential” showing typical road flanked by local shops and parked cars. This image also shows the improvement offered by filtering out decoder noise in the sky regions.

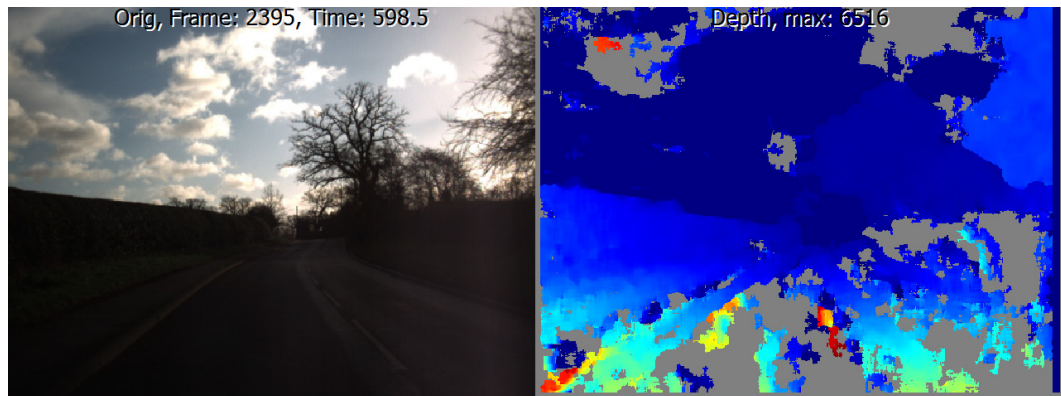


Figure 4.14: Left: image from the left camera, Right: the corresponding depth map. Sample of set “Berkswell automatic”, in this sample the [LIDAR](#) fusion is disabled to demonstrate the poor quality of the stereo data. The red circle on the route map in the dataset description shows where this image was taken.

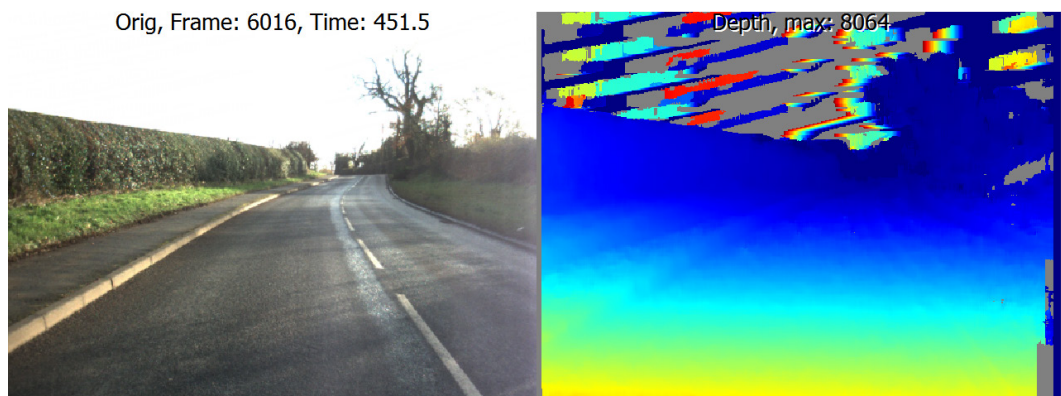


Figure 4.15: Left: image from the left camera, Right: the corresponding depth map. Sample of set “Berkswell manual” showing approximately the same scene as in [Figure 4.14](#) demonstrating the significant improvement seen when switching to manual gain control.

5

IMAGE SEGMENTATION AND ANALYSIS FOR TRAINING

CONTENTS

5.1	Colour and contrast correction	102
5.2	Hybrid image segmentation	105
5.2.1	Step 1: Canny edge detection with region closing	107
5.2.2	Step 2: Depth map segmentation	111
5.2.3	Step 3: Mean-shift texture segmentation	114
5.2.4	Step 4: Segmentation fusion	116
5.2.5	Performance of segmentation fusion method	119
5.3	Choice of colour space representation	122
5.4	Texture and shape properties	128
5.5	Choice of depth augmented descriptors	138
5.5.1	Point cloud orientation	140
5.5.2	Point cloud planarity	143
5.6	Discriminatory analysis with principal component analysis	150
5.7	Conclusion	157

This chapter will outline the major image manipulation steps required to convert raw input images to a set of segment description vectors. First images are segmented using a novel hybrid segmentation scheme which aims to fuse the strengths of edge, texture and depth based segmentation. The goal is to be certain that no segment contains more than a single type of surface.

Because the edge segmentation step, and to some extent the texture segmentation step, depend on clearly defined intensity gradients in the image, the data in all colour channels must be scaled to utilize the full colour space. It is also desirable to eliminate the effect of ambient lighting conditions as much as possible so that classifiers trained on a bright sunny day can still work effectively on images that were captured at dusk. Although it is not possible to eliminate this entirely, the amount of variation can be reduced via image enhancement, discussed in the first section.

After image correction, and segmentation, the collection of segment measurements will be described indicating how each of them contributes to the distinction between different surface types. Next the results of a principal component analysis are given to justify the inclusion of each measurement and to provide an indication of how much unique information each one contributes.

Finally the surface modelling procedure is described showing how the image segmentation is applied to the depth map to extract a 3D point cloud for each segment. Additional information is then collected about the segment using this geometric information, including a novel surface modelling technique to identify the magnitude of the curvature of this physical surface.

5.1 COLOUR AND CONTRAST CORRECTION

As discussed in the introduction, edge detection depends on good image contrast, so raw inputs are first adjusted to improve contrast and hence detail in darker areas. Although the camera Automatic Gain Control ([AGC](#)) was disabled during the recording of the data, some camera parameters still auto-adjusted to compensate for bright skies, leaving the important ground regions underexposed. The first approach to contrast enhancement consisted of taking a histogram of the image and rescaling the intensity of all pixels such

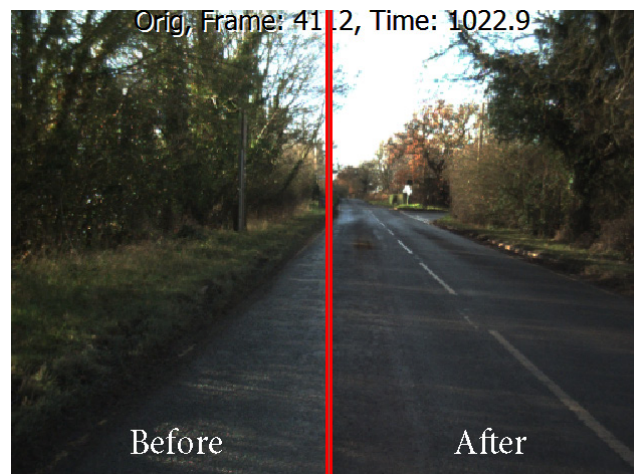


Figure 5.1: Example of ineffective contrast enhancement.

that 1% of pixels were saturated at the highest and lowest levels. The idea was that if all images cover the full range of intensity values then there should be reduced scope for variation.

This simplistic approach did not produce satisfactory results because most images have sky sections which are already saturated at the top end and cover more than 1% of the image. This meant no scaling took place. In fact most images had the dark regions darkened further instead of being enhanced as shown in [Figure 5.1](#).

Looking at work from other authors such as that by Lam [95], Provenzi et al. [150], Rizzi et al. [153] it was decided to implement the grey world assumption to perform contrast and colour correction. The method makes the assumption that the average colour of a collection of independent pixels, composing an image, will be grey. If this assumption holds then the sum of all intensity values divided by the number of pixels in the image for each colour channel should be 127.5 (where 8-bit colour images have intensity values ranging from 0 to 255). If this is not the case then it indicates the image has a colour or brightness bias. To correct the bias each colour channel is rescaled by the ratio between the assumed average (127.5) and its present average:



Figure 5.2: Example of grey-world colour correction.

$$\mathbf{G} = 127.5rc \left(\sum_{i=1}^r \sum_{j=1}^c \mathbf{I}_{i,j} \right)^{-1} \mathbf{I} \quad (5.1)$$

Where \mathbf{G} is the corrected colour channel, r , c are the number of image rows and columns, \mathbf{I} is the original colour channel and $\mathbf{I}_{i,j}$ denotes the channel intensity at row i column j . This is applied to each of the red, green and blue colour channels independently.

Figure 5.2 shows the effect of the grey world correction on two images from the “Berkswell manual set” the left, original image, was captured in overcast conditions and appears overly dark. Notice how the corrected version enhances the contrast of the road surface while saturating the sky further to discard unwanted information. Now look at the right pair of images, the original image was captured in brighter conditions. Notice how the corrected version was not brightened as much as the left hand image since less

correction was needed. Also notice how the brightness and tone is a much closer match in the corrected images, than it was in the original images, showing how the correction has removed some of the image variation under uncertain lighting conditions. Having consistent colours on same surfaces allows the classifier to target a more specific range of values and improve classification performance.

5.2 HYBRID IMAGE SEGMENTATION

Before parts of an image can be classified into their respective types, the image needs to be segmented into smaller regions for analysis. Ideally these regions will be as large as possible without containing more than a single type of content. This content could be anything in the image such as the road, vegetation, pavements, road markings, other vehicles, pedestrians, the sky. If each separate type of content can be separated into its own region then specific tests can be performed on this texture to identify what it is.

Several strategies exist in current literature for performing this segmentation, some of them focus on detecting discontinuities in the image intensity i.e. edges and define these to be the boundaries on different types of content [37], others focus on grouping together areas of similar texture such as the mean-shift segmentation method [34]. The advantage of edge-detection segmentation is that it will separate areas of similar texture if a boundary line exists between them, which is useful in this case when it is necessary to separate the road from the bordering pavement. Texture based segmentation is likely to merge the road and pavement together due to their proximity and similar texture. The advantage of texture, and colour based segmentation, is that it does not over-segment content with hard edges and strong textures, such as foliage and grass. In this work the EDISON mean-shift algorithm [32] was used for the texture segmentation, see [Figure 5.3](#) for an example of the output. Several other mature image segmentation methods were also tested including techniques based on clustering and graph cuts. In terms of grouping regions of similar



Figure 5.3: Example of EDISON mean-shift segmentation.

texture into single regions there was not significant variation in performance. Since EDISON is readily available as a compiled Matlab library and offered decent performance in terms of speed and results it was the chosen method. The parameters were set to preserve edges but, even so, notice how the segmentation has merged a large portion of bare earth and grass on the left of the road with the road itself. If this was the final segmentation it would be impossible to assign separate labels to the road and verge areas. If a system depends on this information to determine which areas are safe to drive on then the verge area should not be included. This problem is also often seen in more urban environments where many tarmac surfaces, such as pavements, are incorrectly merged with the road. This is why the segmentation needs to be more precise which can be achieved using the addition of edge finding.

The image can also be segmented using information not directly present in the picture itself, such as depth map segmentation, which uses discontinuities in the physical spacing of a scene to determine object boundary points e.g. work by Huang et al. [82] and Scaramuzza et al. [156].

To reap the benefits of all of these approaches a composite segmentation scheme has been employed, creating a segmentation using all of the above methods then merging the results together into a simplified and more usable state.

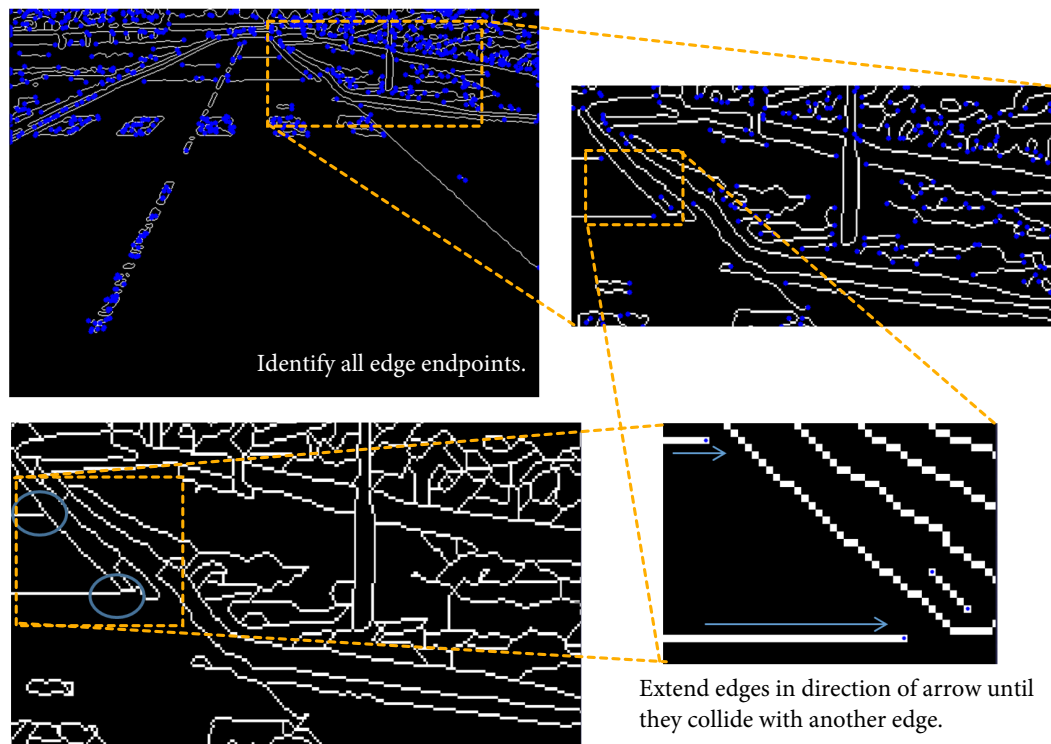


Figure 5.4: Demonstration of enhanced Canny segmentation using a less aggressive threshold than in practice for clarity.

The hybrid segmentation is composed of three parts; the enhanced Canny edge map, the depth segmentation and the EDISON mean-shift segmentation. The fourth step describes the order and method used to combine the three steps into a single segmentation map. See the flowchart on page 16 for clarification. Listing A.2 on page 230 shows a code listing for this procedure.

5.2.1 Step 1: Canny edge detection with region closing

The first segmentation is done using a modified Canny edge detection algorithm [25]. Figure 5.4 shows some of the steps involved on a test image that has been less aggressively segmented than images in the final system to clarify the figures. Actual examples will be

shown later. The basic aim is to take the output of the standard Canny edge detector and then grow the edges until they meet another edge to create closed regions.

- A. The standard Canny edge detection method is applied using sufficiently low thresholds to over-segment the image, at this point it is desirable to have an abundance of weak edges to reduce the chances of missing a true edge. Small segments can be recombined in the final step which is simpler than further segmenting regions that contain multiple textures. [Figure 5.5](#) shows the effect of lowering the threshold, the actual value was empirically determined to be 0.04 for weak edges and 0.07 for strong edges.
- B. The edge lines returned by the Canny detection do not necessarily intersect with each other at their endpoints to create enclosed regions. To rectify this all line segments returned by the Canny detection are inspected to identify end points (defined as points that have only 1 neighbour in their 8 surrounding pixels). In a corrected edge map, that consists of segment boundaries only, no such endpoints should be identified. These points are marked in blue in [Figure 5.4](#).
- C. For each of these endpoints pixels are added to continue the line in the same direction until another edge is encountered. The continuation direction is determined by looking at the position of existing neighbours relative to the endpoint and continuing the line in the opposite direction. This is shown in the lower two portions of [Figure 5.4](#).
- D. If no other edge is encountered within a range of 50 pixels then a secondary search is performed in all 8 (diagonal and orthogonal) directions from the endpoint since it is necessary to have the region enclosed even if the shape is distorted. Pixels already connected to the edge point in question are ignored during this search, see [Figure 5.6](#) on page 110 for an illustration.

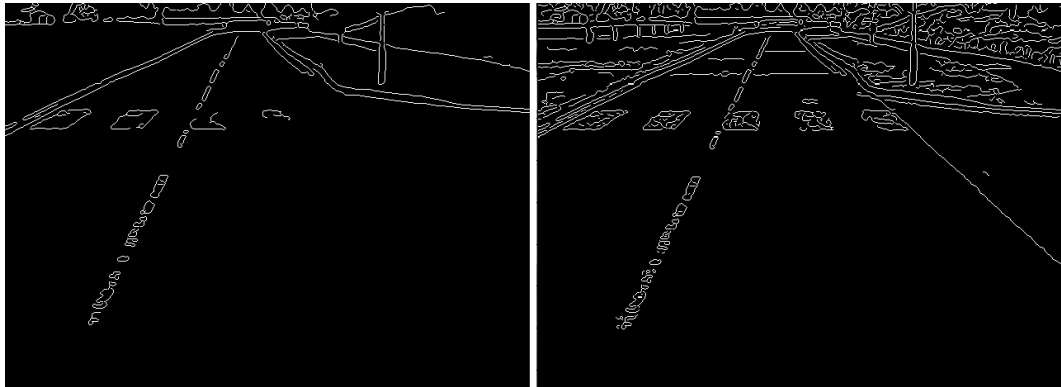


Figure 5.5: Illustration of Canny threshold levels, left image shows a high threshold and the right image shows a lower threshold.

- e. Each of the newly created enclosed regions is assigned a unique integer ID value which is applied to each pixel in that region creating a label image. Then a 3x3 selective mode filter is used to remove remaining edge pixels, including those that do not separate segments. The mode filter is used since pixels are tagged with integer ID's so replacing edge pixels with most common neighbour will preserve the detailed boundaries between segments. Only edge pixels are modified and edge pixels themselves are ignored from the mode count thus no edges can remain. See [Figure 5.8](#) on page 111 for the result.

This method was chosen in preference to using binary image operators, since it preserves the shape of segments more precisely. When image dilation is used to connect edge sections followed by repeated erosion to thin the edges to a single pixel the resulting segments are strongly distorted. Thin regions are reduced to a single line and then, as a result, they get filtered out. [Figure 5.7](#) shows the problem on the sample image above. To create closed regions the Canny edges are dilated by 10 pixels, then binary erosion is performed repeatedly until no further pixels are removed. This fails to close regions that have large gaps and destroys small segments that already existed.

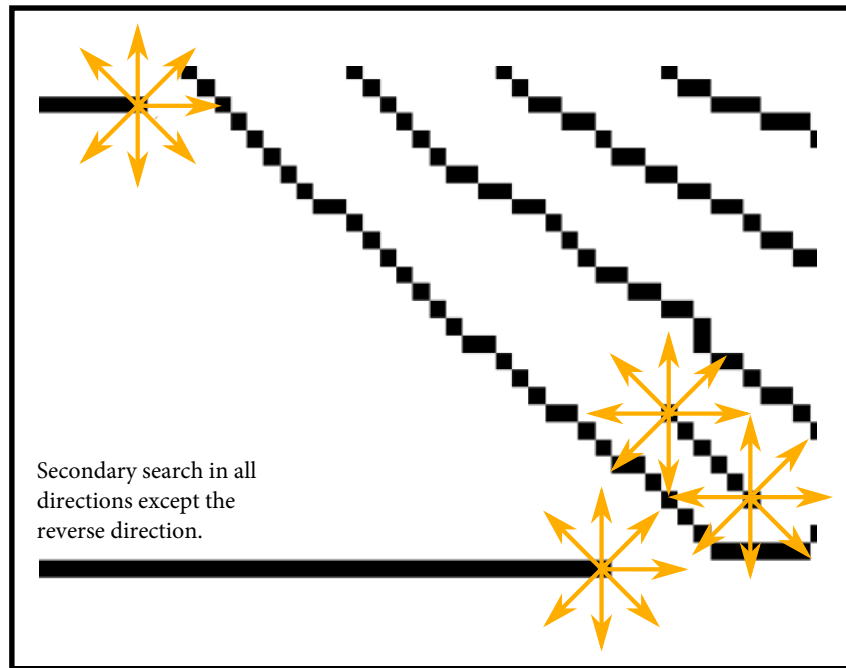


Figure 5.6: If no edge is encountered while growing the edge in its current orientation then a secondary search in all directions is performed to find the nearest edge. This image shows the same clip as the lower right region of [Figure 5.4](#) with inverted colours.

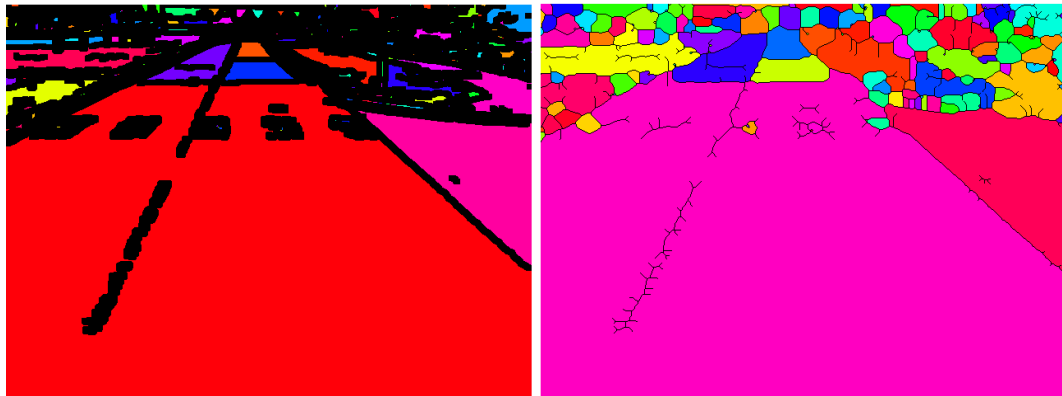


Figure 5.7: Example of binary morphology operations ineffectively segmenting the test image, connected regions are randomly shaded to show the separation between regions.

Compare this to the segmentation achieved with the presented method in [Figure 5.8](#) the shape of segments are well preserved and the image has been segmented such that no regions are incorrectly merged.

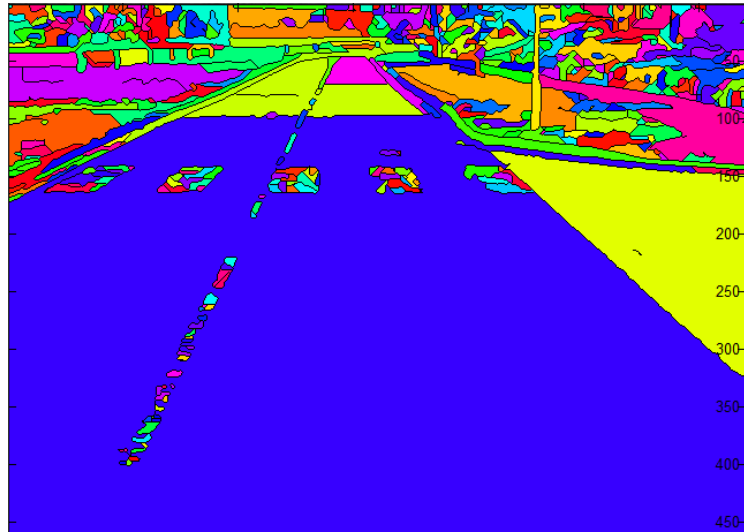


Figure 5.8: Example of the superior segmentation accuracy using presented method. Segments are randomly shaded to show boundaries.

5.2.2 Step 2: Depth map segmentation

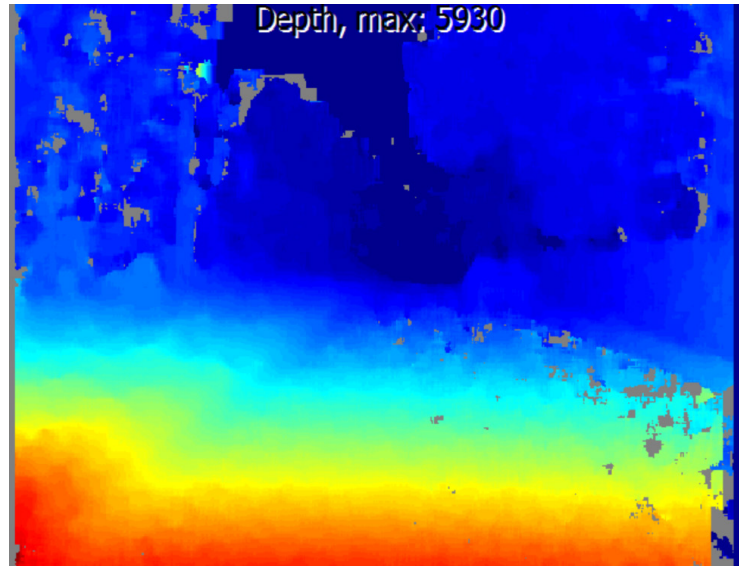


Figure 5.9: Example depth map to demonstrate depth segmentation.

The depth map of the scene can provide strong edges in the sense that whenever there is a physical discontinuity in the scene there must always be a segment boundary in

the same place. The problem is that, in general, the only parts of a scene that cause strong discontinuities are those where a nearby object is occluding an object further away. This means that this segmentation method will be entirely ineffective at separating surfaces that run smoothly into each other, even when they have dissimilar appearance. For example a road surface will run smoothly into a pavement surface without creating a significant disturbance in the depth map. For this reason, the depth map segmentation is most useful at identifying boundaries between objects and background surfaces, such as a car against a road. [Figure 5.9](#) shows the example depth map for the following discussion.

Initially this step was incorporated because it offered the potential to create indisputable boundaries between physical objects, however, after observing the effectiveness of the edge and texture segmentation, it was noted that there were not many cases when the boundaries marked by the depth segmentation were not already found in the Canny step. For this reason it should be considered the least important step and could be dropped if there was need to slightly reduce the computational burden, for example. Since there are cases when it does identify a boundary, which was missed by the other steps, and because the computational overhead of this step is in the order of 5–10ms it was decided to retain the step for the slight improvement in accuracy it brings.

The actual method for segmenting the depth map is the same as the method for segmenting the image in step 1, with some modification. First the disparity data is rescaled to the range $[0, 1]$ in order to be passed through the Canny filter. Next the same steps from A–D, as described above, are followed with the same thresholds. Because the disparity image is so smooth the resulting edge-map is sparse and even after the 8—direction search many lines are still not closed into self contained segments. For this reason, in the final step, E is not used leaving some enclosed segments and some unconnected lines.

At this stage a sparse edge-map is obtained as seen in the left part of [Figure 5.10](#) but due to the missing values in the disparity map, edges are marked around any region

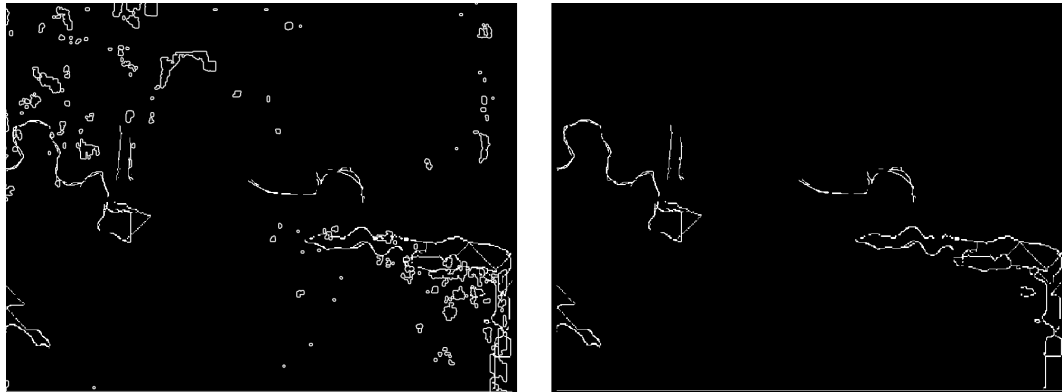


Figure 5.10: Removal of noise from depth segmentation, false edges caused by missing data in the depth map are removed by filtering out edge pixels that neighbour an undefined region in the depth map. The left image shows the edge map before and the right image shows the outcome.

that is missing depth information in the disparity map. This noise is easily removed by using a filter which eliminates any edge pixels from the edge-map where a neighbouring pixel, in the same position in the disparity map, contains missing data. See the right side of [Figure 5.10](#) for an example of this filter. Notice that the grey regions in [Figure 5.9](#) (indicating missing depth information) do not create corresponding edges in the output, due to the cleaning filter.

All that is left, after removing false edges on the borders of missing regions, is some of the outline of the car and bushes on the left. Some noise caused by disruption in the depth-map is still present in the lower right, however this is generally lost in step 4 when small regions (which is generally how this type of noise manifests itself) are merged. In order to keep these important edges, and avoid having them filtered out by the extraneous line filter, this edge-map is immediately merged with the edge-map obtained from step 1 which will generally overlap existing edges, or split existing segments into smaller ones. In cases where lines from this step do not intersect with existing lines in step 1, they will still be intentionally filtered out since lines which do not intersect are generally noise. See [Figure 5.11](#) for an example of this, the green lines show the resulting depth map edges

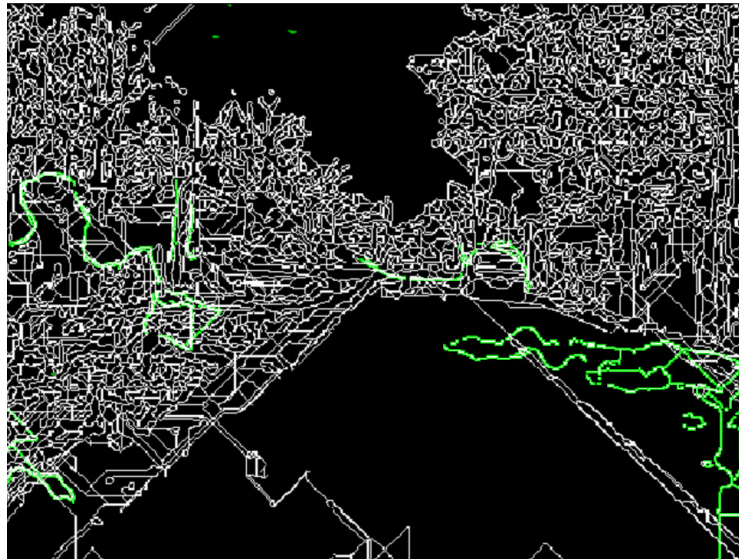


Figure 5.11: Merging of depth segmentation with edge segmentation, the green lines show the depth map segmentation intersecting existing lines in the edge map segmentation. This image shows the result before filtering out unconnected lines which will remove the majority of the green lines in the lower right corner.

merged with the result from step 1, notice how in most cases the lines overlap existing edges.

5.2.3 Step 3: Mean-shift texture segmentation

This step is basically an “off the shelf” mean-shift image segmentation step. Using the colour corrected image and the EDISON library for Matlab [32] it was a simple case of tweaking the parameters to get the right balance between fidelity and segment size. Since the purpose of this step is to recombine small segments from the edge detection segmentation step the aim, when adjusting the parameters, was to get a small number of large segments which combine roughly textured regions such as trees and hedges into larger, more manageable, regions. The output of this step alone would be ideal if not for the problem of merging similar areas across clear line boundaries. The package settings essentially allow the user to trade-off between small, detailed, segments or larger segments covering greater changes in texture. This means that it is not possible to get the

Table 5.1: Parameters used in EDISON package.

PARAMETER	VALUE
SpatialBandWidth	9
GradientWindowRadius	5
MixtureParameter	0.3
RangeBandWidth	6.5

desired behaviour of smoothing out rough textures, whilst also maintaining boundaries between similar textures, without using this hybrid technique. Even though the EDISON algorithm does use a synergy of edge detection and mean-shift approaches it does not provide the level of fine tuning required to get both the desired effects, on different parts of the image, at the same time.

With these considerations taken into account the final parameters were chosen to break the image into 300–500 segments, as opposed to the thousands produced by the edge detection step. This step should find and merge regions of similar texture which can later be used to replace the over segmentation produced in step 1, where the regions are unworkably small. See [Figure 5.3](#) or the right hand column of [Figure 5.12](#) on the following page for an example input and output.

For those interested in reproducing the same results as seen here the exact values used to run the EDISON step are listed in [Table 5.1](#). Unlisted parameters are left at default, those that were changed were modified empirically to encourage the merging of roughly textured areas, at the expense of splitting similar regions. Explanations for these parameters can be found in the documentation for EDISON which can be found online at: <http://coewww.rutgers.edu/riul/research/code/EDISON/>.

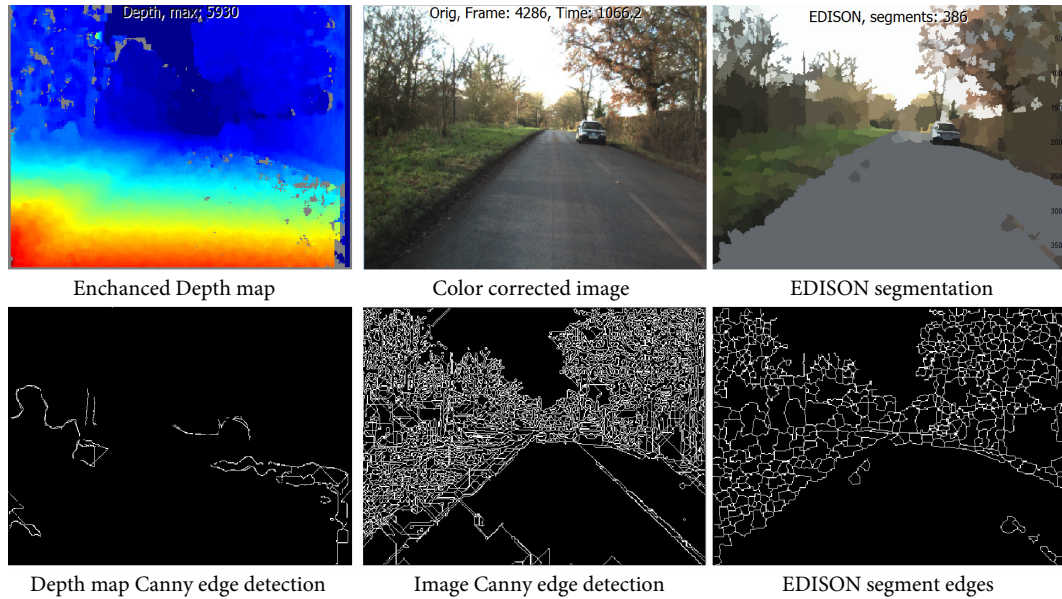


Figure 5.12: Outputs from the three segmentation fusion steps, from left to right, depth map segmentation, edge segmentation and texture segmentation.

5.2.4 Step 4: Segmentation fusion

The final step in the segmentation fusion is the merging of the previous three steps in such a way that preserves the positive aspects of each while removing the negative aspects. To summarise, step 1 (edge segmentation) has the desired detail but tends to over-segment in strongly textured regions. Step 2 (depth map segmentation) provides strong, high importance, edges but alone does not create closed regions. Step 3 (mean-shift segmentation) combines strongly textured regions, but also merges similarly textured regions which should be separated by an edge. [Figure 5.12](#) shows the input and output of the three steps.

As discussed in step 2 the results from steps 1 and 2 are immediately merged into a combined edge map as shown in [Figure 5.11](#) on page 114 which only leaves the combination of this result (which will now be referred to as the edge+depth segmentation) and the mean-shift segmentation.

The aim here is to replace small regions in the edge+depth segmentation with the larger texture combining regions of the mean-shift segmentation. The definition of a small region depends on the resolution of the images and how large a region needs to be in order to calculate meaningful statistical information on its appearance. In this case it was determined that over 95% of regions would contain enough depth data to complete robust plane fitting (see next section) if the segment size is greater than 80 pixels, assuming the segment in question is closer than 30m to the camera. At greater distances the depth information becomes too sparse without using segments that are too large to maintain object shape detail.

Each closed region in the edge+depth segmentation and the mean-shift segmentation is assigned a unique (to both images) integer ID label. The label is applied to each pixel in the segment creating a label image. Then regions in the edge+depth segmentation are filtered to isolate those with an area smaller than the 80 pixel threshold as shown in [Figure 5.13](#). Next, for each of those regions identified, the label ID for that region will be replaced with the most common ID of the pixels in the corresponding position of the mean-shift segmentation. Generally, because the regions are smaller than those they are being replaced with, there is only one ID found in pixels from the same area in the mean-shift however, in the case where the region crosses a boundary in the mean-shift segmentation, the most common ID is taken. This means that the shape of the original segment is preserved i.e. does not distort desirable accurate edges found in step 1. Large regions never get replaced, therefore when two areas with similar textures are separated by an edge, they never revert to the mean-shift segment shape.

When this is done there will be many clusters of regions in the edge+depth segmentation with identical ID's. The next step is to remove edges between regions with identical ID's allowing the previously small regions to merge into one larger region with the segment outline composed of the outer edges of the original segments, not those found by the mean-shift. The result can be seen in the right side of [Figure 5.13](#) after all merging is done

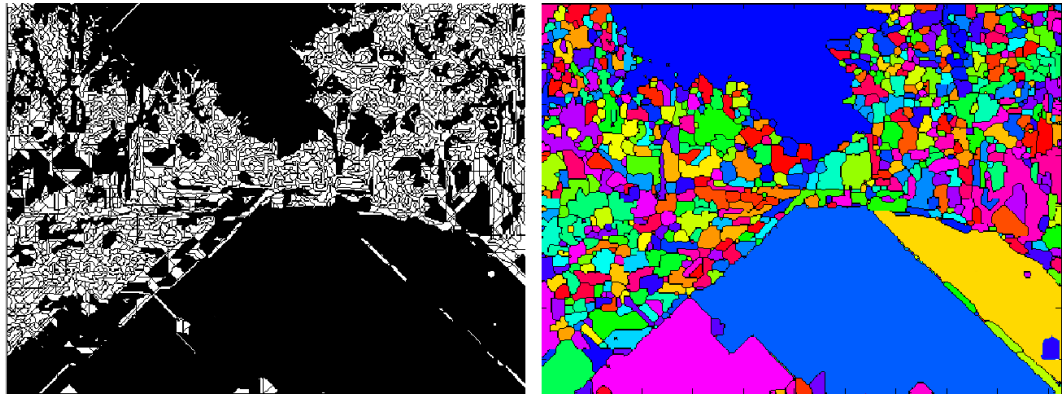


Figure 5.13: Merging of small regions from edge segmentation, the left image shows, in white, all regions with area less than 80 pixels, the right image shows the result after merging them together using the the mean-shift segmentation as a guide. The regions are randomly shaded to show boundaries.

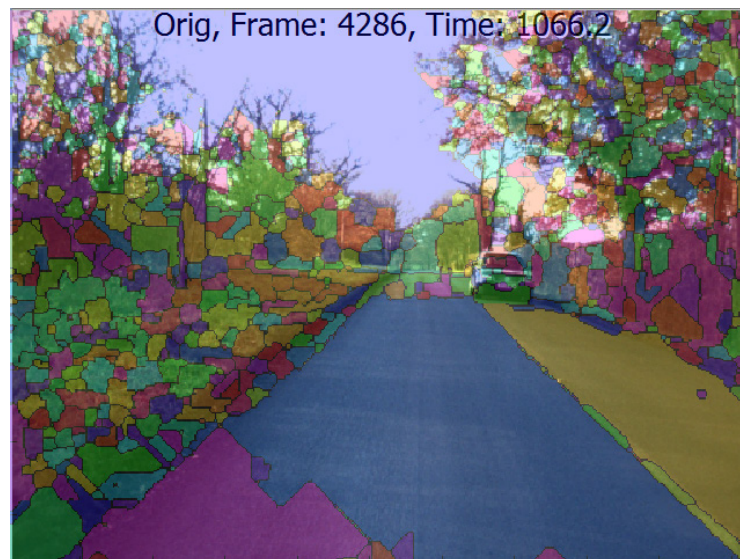


Figure 5.14: Output of segmentation fusion on sample 1.

and a final filter to remove edges that do not enclose a region is applied. See [Figure 5.14](#) on this page for the resulting segmentation overlaid on the original image and [Figure 5.15](#) for another example on the image used to illustrate step 1.

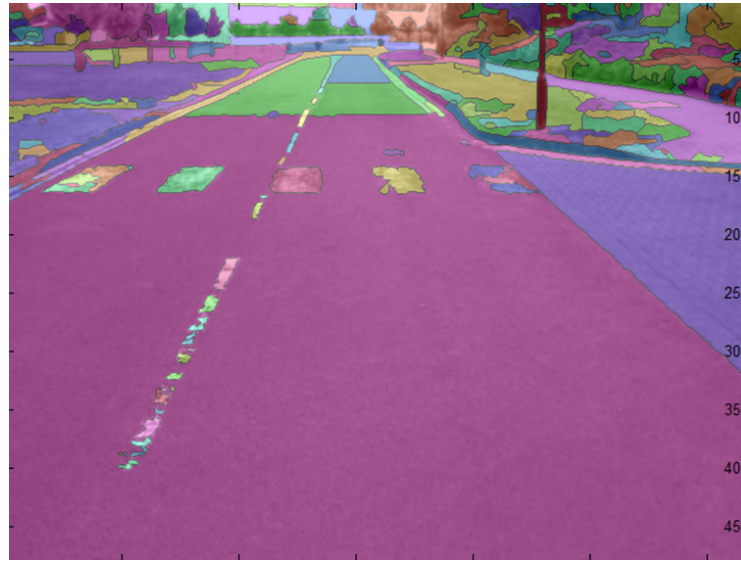


Figure 5.15: Output of segmentation fusion on sample 2.

5.2.5 Performance of segmentation fusion method

The purpose of the image segmentation is to break the picture into regions of only one type. Ideally this condition will be met using the minimum required number of segments to reduce computation time of segment descriptors. However, since the image content is not known, it is difficult to know how aggressively to segment an image. It is an acceptable compromise to have more segments than necessary providing no segment contains more than one class of surface since this is the top priority. If a segment spans several content types then the types cannot be individually labelled. Figure 5.16 shows an example of this, the bush and the lamppost in the white circle cannot be separately labelled because a segment spans them both.

If the segments are too small then it creates two problems. First the texture measurements will not be statistically significant due to small sample size (too few pixels). Secondly the surface reconstruction will behave erratically due to insufficient data points. As discussed in step 4 of this section an appropriate threshold is 80 pixels.

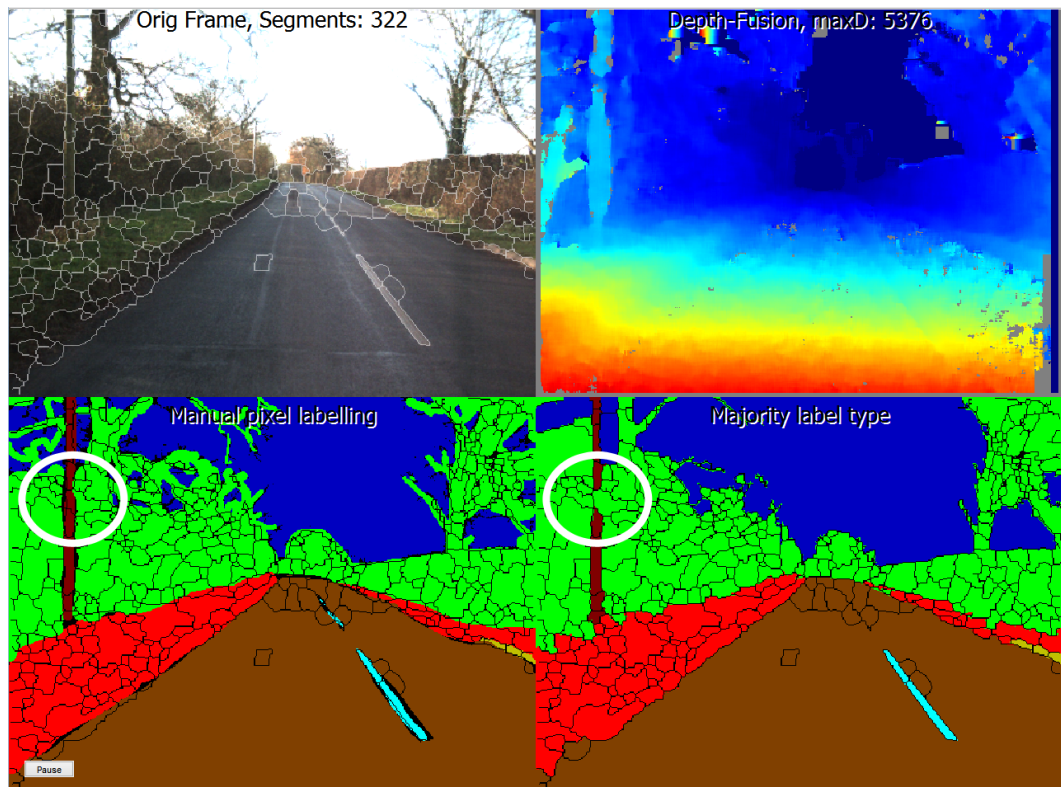


Figure 5.16: Evaluation of segmentation performance, top-left shows original image overlaid with automatic segment boundaries, top-right shows the corresponding depth map, lower-left shows manual image segmentation with same overlay, lower-right shows the automatic segments labelled with the manual label that composes the majority of their area. The white circle shows an example of where the segmentation has failed to separate two types of content that were separated in the manual segmentation.

With this objective in mind the proposed segmentation method was evaluated against manually segmented images. A group of five images from the “Berkswell Manual” dataset (see [Section 4.3.4](#)) were manually segmented. The manual segmentation was performed such that regions are as large as possible without containing more than one type of scenery as described. The test is to see how many of the automatically generated segments, using the proposed method, cover more than one of the manually defined segments. Secondly the number of resulting segments below the size threshold is recorded. This test was performed using the proposed fusion segmentation, the edge segmentation alone (step 1) and the EDISON segmentation alone (step 3).

Table 5.2: Summary of segmentation performance test, the “mixed” column shows the percentage of segments that contained more than one content type, the “< 80px” column shows the percentage of segments below size threshold. In this test lower scores are better.

IMAGE	FUSION		EDGE		EDISON	
	MIXED	< 80PX	MIXED	< 80PX	MIXED	< 80PX
1	8.3%	1.6%	0.9%	84.3%	21.1%	1.2%
2	9.2%	1.9%	1.1%	85.8%	22.2%	1.5%
3	8.2%	1.7%	0.8%	81.1%	25.7%	1.3%
4	11.2%	2.1%	1.2%	88.5%	31.2%	1.6%
5	7.9%	1.6%	0.8%	83.1%	21.4%	1.1%
MEAN	9.0%	1.8%	0.9%	84.6%	24.3%	1.3%

A segment is considered to be spanning more than one label type if its majority content is less than 80% of its total area. i.e. if more than 20% is of another (or several other) types then it is considered a mixed segment. This was necessary due to the imprecise hand-drawn manual image labelling which does not adhere exactly to content boundaries.

Table 5.2 shows the results from the experiment. Figure 5.16 shows image 1 segmented with the fusion method. It can be seen that the majority of the 8.3% of mixed segments are located along the boundary between the grass and the hedge. With this type of content transition the boundaries are often fuzzy as grass and hedges do not have well defined edges. Errors like this are unimportant as they are caused by a disagreement in boundary location instead of actually merging two distinct content types. Errors such as the mixing of the lamppost and hedge (circled in white) are considered true mistakes that will degrade the quality of classification and training data.

It can be seen in Table 5.2 that overall the fusion method provides the best combination of low mixed segment rates and almost no undersized segments. The edge segmentation regions are so small that they almost never span two labels and those that do are generally due to boundary disagreement. This seems like an effective segmentation method, however the segments are so small, and numerous, (over 1000 segments) that any texture and

shape analysis would be time consuming and inaccurate. [Figure 5.13](#) on page 118 gives an example of how many segments under 80px exist in a typical segmentation. Finally the EDISON method, as discussed, was tuned to provide large segments so as expected it has the lowest rate of segments that are too small. The problem with it is that the segments often span content types as seen by the high mixed rates.

Notice how the fraction of segments that are too small are almost identical in the fusion and EDISON method. This is because the EDISON is used directly to eliminate small segments in the edge segmentation. In general the segment size in the final fusion segmentation will approximately match the EDISON segment. This is because the small segments that fall inside a single EDISON segment are merged together into a similar shape. This does not happen only when there are large uninterrupted regions in the edge detection step.

It is not a case of comparing the performance of the three segmentations – each of them is intentionally tuned to have a particular strength. The experiment is to show that the fusion step retains the strengths of all but does not share their limitations.

5.3 CHOICE OF COLOUR SPACE REPRESENTATION

An intensity map (or grey scale version) of the image is created using the per-pixel weighting of $0.2989R + 0.5870G + 0.1140B$ [19] which is used for texture analysis measurements. The colour channel weighting is the default setting used by Matlab's internal colour to grey scale conversion function it is intended to match the sensitivity of the human eye to each colour, this gives the grey scale images a more realistic contrast (to our perception) compared to using a straight average. This intensity map is used for texture descriptors and contrast measures described in the next section.

The intensity image discards all colour information but the separate colour channels contain useful information that should be incorporated into the feature descriptors. One way of representing this colour information is to include the average amount of red, blue and green in each segment, the combination of these three values specifies the particular colour of that segment. This approach was used in preliminary tests but it was found that these three measures across an entire dataset of segment descriptors, were close to 100% linearly dependant i.e. the correlation of the average red, blue and green components for all segments was greater than 99%. This was revealed in a Principal Component Analysis (PCA) analysis of the dataset similar to the one described in [Section 5.6](#).

The problem is that the three components together also represent the intensity of that colour i.e any dark colour will have low values in all channels and any bright colour will have a high value in all channels. The deviation between them is small, but specifies the important perceived colour. This way of representing colour information is not invariant to lighting conditions and is not well suited to machine learning techniques which aim to separate classes based on the distribution of feature values. To demonstrate this problem see [Figure 5.17](#), it shows a typical image divided into its three component colour planes. Notice how, although each plane represents the intensity of a different colour, the overall appearance of each is close to identical due to the fact that the brightness of that colour must be the average of the three.

To address this issue, images are converted into Hue, Saturation, Value (HSV) colour space as used by a number of image processing researchers such as Sural et al. [168]. In this colour space the perceived colour is separated from the brightness and intensity. The benefit is that colours in the image gain independence from their illumination which is of great importance in this area of research as it allows a patch of road in direct sunlight to have the same colour value as a patch that is in the shade, thus allowing the classifier to cope with variable lighting intensity within scenes. This colour is represented by the hue, which instead of ranging from one colour to another, wraps round in a circular

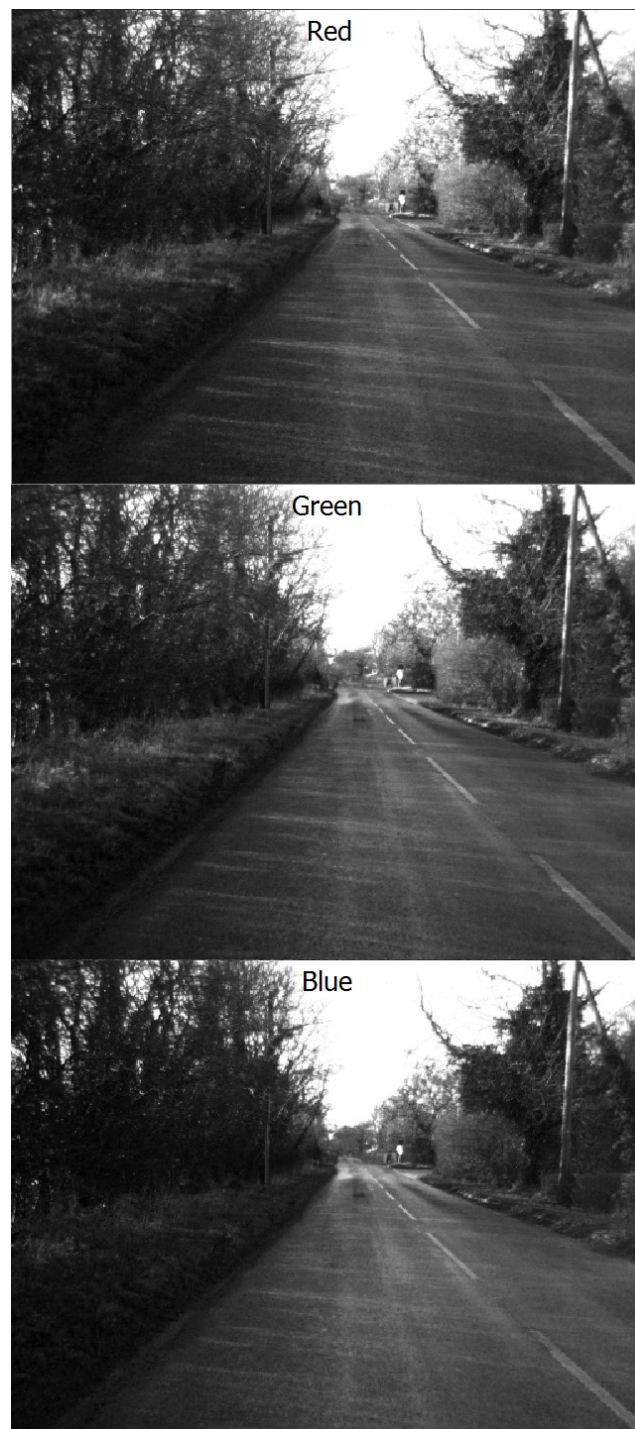


Figure 5.17: Example of RGB colour space, each image represents a single colour channel. The intensity is represented in grey scale for comparison, brighter shades indicate greater intensity in that channel.

fashion such that zero hue is the same colour as maximum hue and all other colours lie in-between. A vivid red and a washed-out red will both have the same hue, but will be distinguished by their saturation value which describes the “greyness” of a colour. In this colour space neutral grey colours will have zero saturation and vivid colours will have maximum saturation. Finally the *value* represents the brightness of the colour and contains similar information to the intensity map calculated from the Red, Green, Blue (RGB) channels.

The conversion from RGB to HSV is defined by the authors of the idea (Joblove and Greenberg [86]) using the following process. First the intermediate variables are defined as:

$$\begin{aligned} M &= \max(R, G, B) \\ m &= \min(R, G, B) \\ \Delta &= M - m \end{aligned} \tag{5.2}$$

where (R, G, B) are the intensity values of each colour channel scaled to lie in the range [0, 1]. Next the HSV values are defined as follows:

$$H = \begin{cases} 60^\circ \left(\frac{G-B}{\Delta} \bmod 6 \right) & , M = R \\ 60^\circ \left(\frac{B-R}{\Delta} + 2 \right) & , M = G \\ 60^\circ \left(\frac{R-G}{\Delta} + 4 \right) & , M = B \end{cases} \quad (5.3)$$

$$S = \begin{cases} 0 & , \Delta = 0 \\ \frac{\Delta}{M} & , \Delta \neq 0 \end{cases} \quad (5.4)$$

$$V = M \quad (5.5)$$

This gives a hue value in degrees which is used to map the colour space onto a hexagonal cylinder, most commonly used in computer art packages. However, for this application, it is more appropriate to have all values fall in the range $[0, 1]$ so the hue will be mapped to a circular cylinder instead using the conversion [72]:

$$H = \text{atan2} \left(\frac{\sqrt{3}}{2}(G - B), \frac{1}{2}(2R - G - B) \right) \quad (5.6)$$

Consider [Figure 5.18](#) and see how different colours in the image are strongly differentiated from each other by the hue and the lighting patterns seen in [Figure 5.17](#) are significantly reduced. All of the road section, which is a neutral grey colour, has low saturation and the coloured grass has high saturation. The hue has a separate value for each different colour in the scene, and with help from the grey-world colour correction discussed in the first section, allows patches of surfaces from different images to be colour matched more robustly.

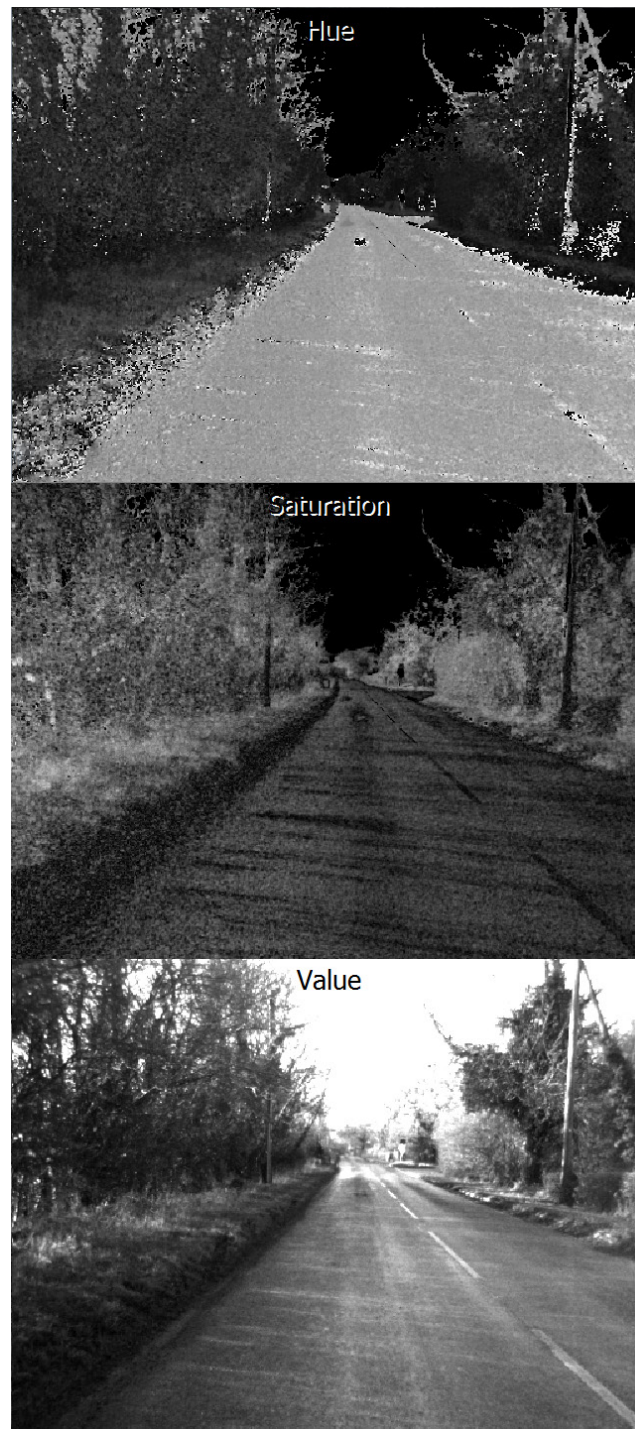


Figure 5.18: Example of [HSV](#) colour space. Brightness indicates the value of the attribute where black is minimum and white is maximum.

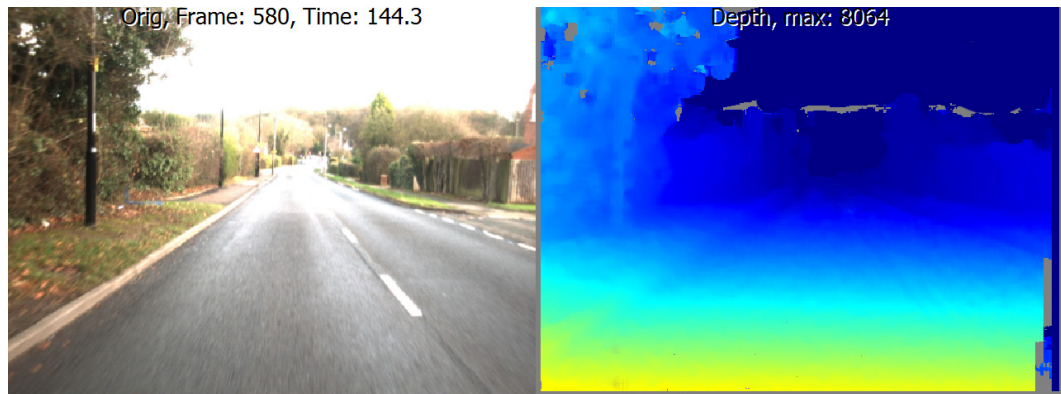


Figure 5.19: Sample image to demonstrate texture descriptors, on the right is the corresponding depth map after [LIDAR](#) fusion.

Instead of using [RGB](#) as feature descriptors the mean hue and saturation are used as they provide uncorrelated representations of the colour content.

5.4 TEXTURE AND SHAPE PROPERTIES

After the image has been segmented, each of the separate regions are analysed in turn to determine characteristic properties which allows the classifier to distinguish one type of surface from another. [Figure 5.20](#) to [Figure 5.35](#) show colour coded responses to each of the values being measured on a typical image shown in [Figure 5.19](#). Unless otherwise stated all intensity measurements are normalised to the range $[0, 1]$.

HUE INFORMATION As discussed in the previous subsection the hue value provides a colour identifier which is significantly less variant under different lighting conditions than colours represented in the [RGB](#) colour space. To encode the hue information for all pixels in the region, the mean and standard deviation of the values was calculated. The mean gives the average colour in the region and the standard deviation describes the extent of colour distribution within the segment. See [Figure 5.20](#) for a visualisation.

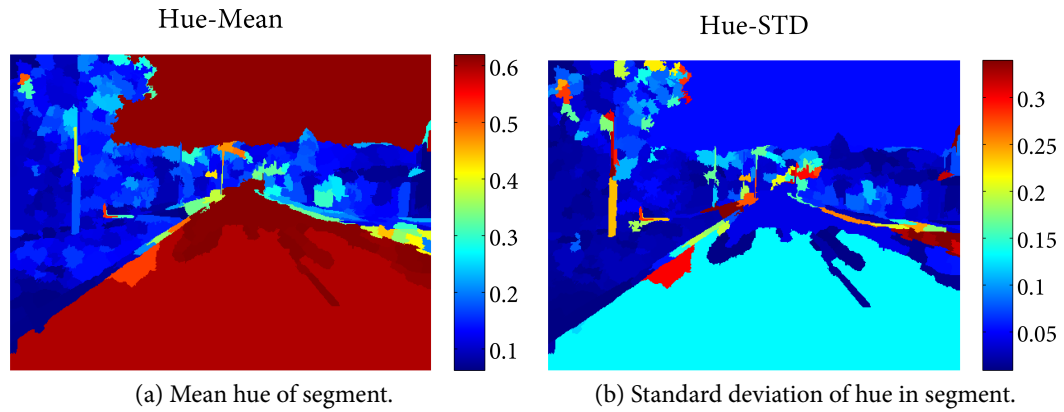


Figure 5.20: Colour coded magnitude maps showing hue mean and hue STD.

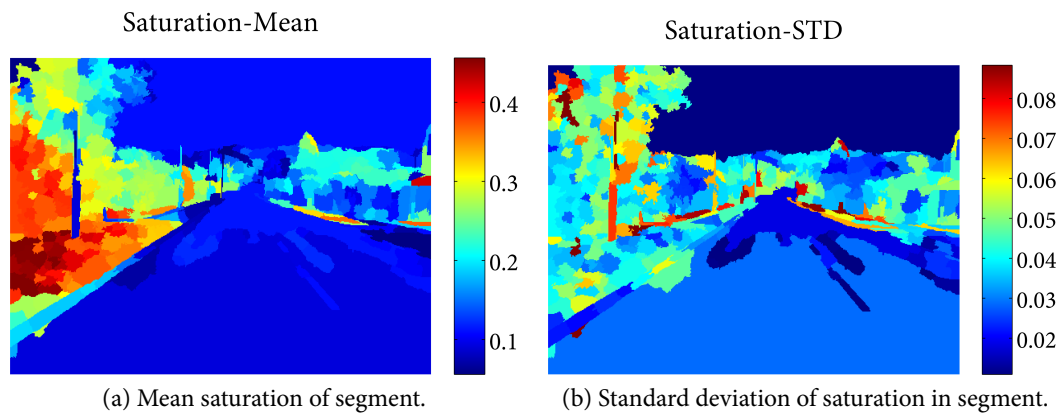


Figure 5.21: Colour coded magnitude maps showing saturation mean and saturation STD.

SATURATION INFORMATION Saturation is included to support the hue information, while the hue describes the specific colour, the saturation determines the intensity of that colour. It is subject to change under different lighting conditions, but allows distinction between different tints of green, for example, that of grass and tree foliage. Again the set of values is summarised by taking the mean and standard deviation. See [Figure 5.21](#) for a visualisation.

RELATIVE AVERAGE INTENSITY This is the numerical mean of all pixel intensity values in the segment compared to the mean intensity of the entire image. The value is divided by the image average intensity in an attempt to further compensate

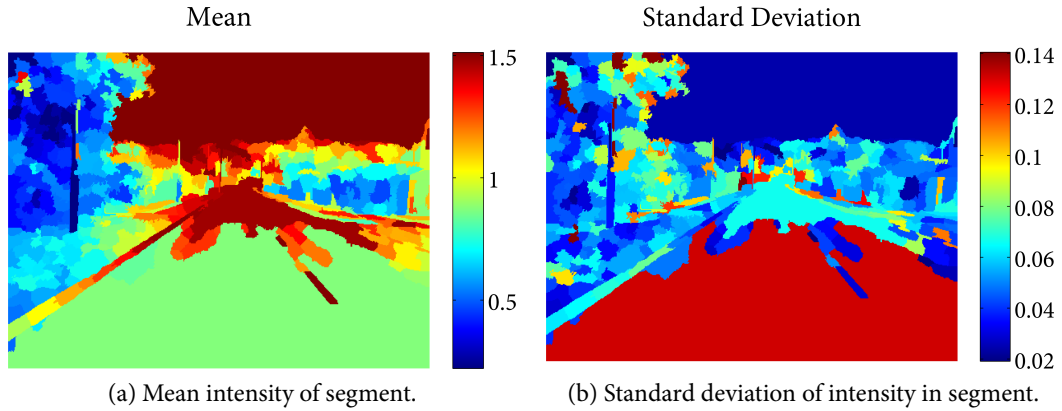


Figure 5.22: Colour coded magnitude maps showing intensity mean and intensity STD.

for the automatic gain control applied by the camera hardware. The idea is that segments of similar colour will have a similar intensity relative to the image average across a wide range of lighting and gain conditions. This measure will be particularly useful for identifying zones that have consistent colour, no matter where they are found such as roads. Conversely it will be less useful for zones that have variable colour, such as vegetation and vehicles. See [Figure 5.22 \(a\)](#) for a visualisation.

STANDARD DEVIATION This represents how much variation, or deviation from the mean intensity, exists in the segment. In the context of texture analysis it is a way of quantifying the granularity, or noise content, within each segment.

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2} \quad (5.7)$$

Where s is the standard deviation, N is the number of samples (in this case the number of pixels in the segment), \bar{x} is the mean intensity of all x_i which are the pixel intensities. Unlike entropy, this will result in a high value for regions with repeating patterns, so together these measurements will provide a way to identify segments meeting this description. See [Figure 5.22 \(b\)](#) for a visualisation.

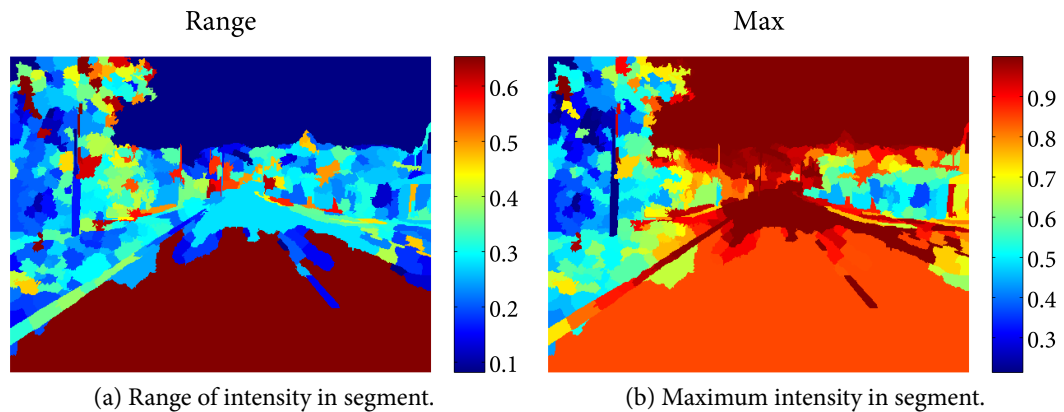


Figure 5.23: Colour coded magnitude maps showing intensity range and maximum intensity.

RANGE The range of the intensity values of pixels in the segment, again normalised by the average intensity of the image to further compensate for variable lighting conditions. The purpose of this is to work in combination with the entropy measure to identify segments with gradients. Segments with high range and low entropy are likely to contain smooth gradients (i.e. the intensity transitions smoothly from one level to another without sharp discontinuities). Segments with low range and high entropy contain low level noise, or random texture. Segments with low range and low entropy are likely to be plain, smooth, untextured areas such as the sky. This provides useful discriminations when combined with texture measures, but may not give the desired results on surfaces with bright spots, or sudden discontinuities, such as shiny surfaces with reflection spots, although occurrences of this should be low due to the design of the segmentation. See [Figure 5.23](#) (a) for a visualisation.

MAXIMUM INTENSITY This is the brightest pixel (or pixels) within the segment, normalised against the average intensity of the image. As with the relative average intensity, this value gives information about the brightness of a segment compared to the entire image. While this will give similar variation to the relative intensity, it may help identify reflective surfaces which will have bright spots among a darker area. i.e. segments with low relative intensity but a high maximum value may indic-

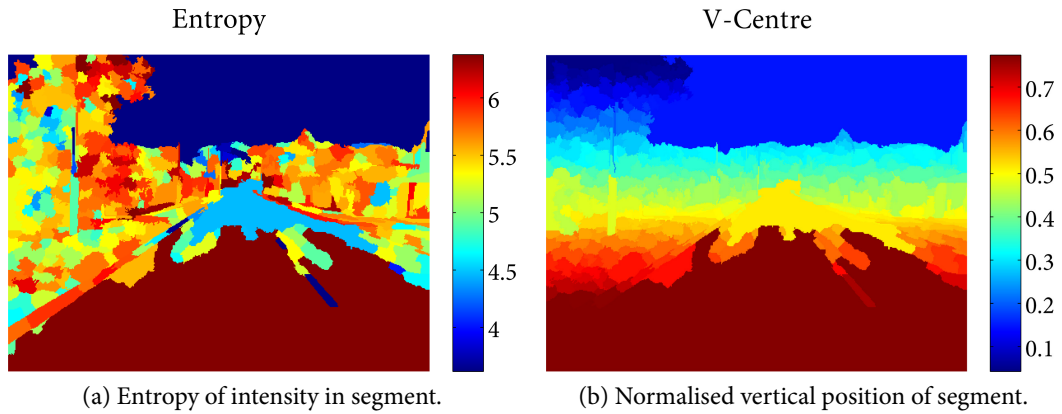


Figure 5.24: Colour coded magnitude maps showing entropy and vertical position.

ate the presence of a glass or polished metal object. [PCA](#) analysis will reveal if this measure offers any benefits over using only relative intensity. This measure may introduce high error rates in images that have saturated regions i.e. where many segments have regions that are over exposed they would all be assigned equal score for this measurement creating incorrect connections and loss of discrimination. See [Figure 5.23 \(b\)](#) for a visualisation.

PIXEL ENTROPY Entropy is a statistical measure of randomness that can be used to characterize the texture of the input image. Since entropy is not easily described with words its definition is:

$$H(X) = - \sum_{i=1}^{256} p(i) \log_2 p(i) \quad (5.8)$$

Where $H(X)$ is the entropy measured in bits, X is the intensity of each pixel in the segment and p is the occurrence count of each intensity level in X (there are 256 levels in an 8 bit grey-scale image) normalised by the pixel count. Simply put, it allows discrimination between very grainy random regions and regions where the texture is smooth, or repeated, and predictable i.e. regions that are not smooth may still have a low entropy if there is a repeating pattern. This measure is sensitive to

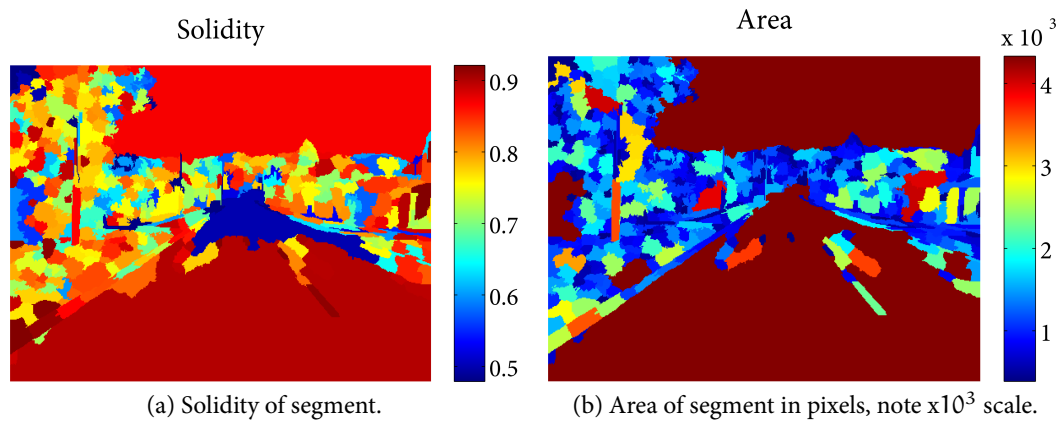


Figure 5.25: Colour coded magnitude maps showing solidity and area of segments.

noise so may give higher than expected results in dark images, due to the increased gain used by the camera. See [Figure 5.24 \(a\)](#) for a visualisation.

VERTICAL POSITION The normalised vertical position of the segment's weighted centre. This is particularly useful for identifying objects that are always found in a certain horizontal band of the image, for example the sky segments will invariably have a greater vertical position relative to any other segment. The absolute value of the vertical position is not important, only the value compared to other segments. As the camera is mounted in a fixed position, the horizon point in the image should be the same across all frames, excluding cases when travelling up steep hills or on uneven ground. This should allow discrimination between, for example, grass and tree foliage. While in most aspects they appear similar they can be differentiated by noting that grass is most commonly in the lower half and tree foliage in the upper half of the image. See [Figure 5.24 \(b\)](#) for a visualisation.

SOLIDITY This is the ratio of a segment's area in pixels to the area of its convex hull. The convex hull is the minimal convex polygon that will contain the segment which can be easily visualised by imagining a rubber band is wrapped tightly around the shape. If the ratio of area, to convex hull area, is high then it indicates that

the object has smooth straight, or curved, edges on all sides; for example it may closely resemble a rectangle. This is the expected shape of the road, pavement and most other man-made objects. On the other hand natural objects, such as trees and other vegetation, will have rough and furrowed edges giving a low solidity measure, providing useful discrimination value. This measurement, in its current implementation, is the most computationally expensive and would at this time make real-time processing difficult. However, with ever improving hardware, it was decided this should not be a reason to exclude it. See [Figure 5.25 \(a\)](#) for a visualisation.

TOTAL AREA This is simply the area in pixels of the region. The idea is that some types of zone such as roads will have large uninterrupted regions spanning large sections of the image which will strongly distinguish them from other zones. Vegetation, for example, will be segmented as a group of small regions due to the high density of hard edges and discontinuities. This measurement may not be effective in cases where zones, normally expected to have large areas, are heavily obscured, for example, if a road is mostly covered in cars. It will also produce non ideal results in conditions where there are hard shadows on the surface as this will cause it to be broken into a number of smaller regions. If large regions are broken down into sufficiently small areas, then it may begin to introduce confusion with zones that always have small regions. See [Figure 5.25 \(b\)](#) for a visualisation.

HORIZONTAL DISTANCE FROM CENTRE The normalised horizontal position of the segment's weighted centre relative to the middle of the image. The idea of this measurement is that it will identify objects that are expected to be near the edges of the view port. For example pavements and roads have similar colour and texture. One of the ways they can be distinguished is by noting that the road is generally near the centre of the image, whereas the pavements are generally closer to the edges. The measurement is taken from the centre of the image so that left and right

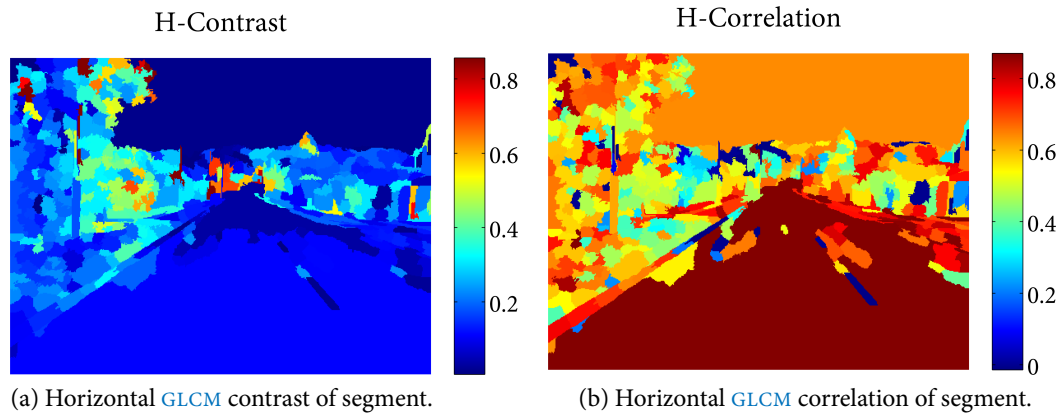


Figure 5.26: Colour coded magnitude maps showing horizontal GLCM contrast and correlation of segment.

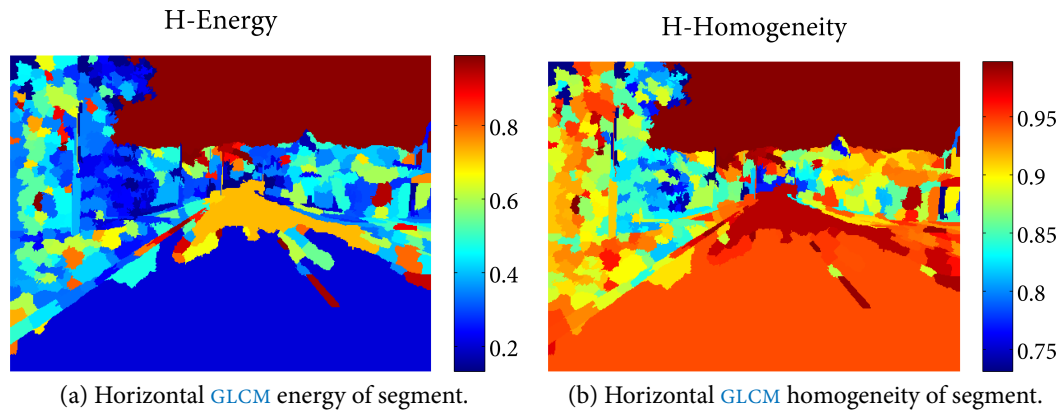


Figure 5.27: Colour coded magnitude maps showing horizontal GLCM homogeneity and energy.

positions are reflected i.e. being on the right edge is equivalent to being on the left. The problem with this measurement is that it does not apply when making sharp turns, or when driving on roads with many lanes.

GREY-LEVEL CO-OCCURRENCE MATRIX ANALYSIS (GLCM) Also known as the grey-level spatial dependence matrix, this is a texture analysis tool that attempts to classify, or characterise, the pattern or texture in a region. Based on usage in current literature this method has previously been applied to geoscientific study (land usage surveys) [169, 36, 109], meteorology [172, 98, 30] and industrial inspection [175, 173, 57]. The algorithm first rescales the intensity of the image to 8 evenly

spaced levels (down from 256 levels) such that pixel values of 1 – 32, in the original image map, to a value of 1 in the scaled version and values 33 – 64 map to 2 and so on up to level 8 in the scaled image. After this the algorithm compares each pair of horizontally side-by-side pixels and records how many times each pair of intensity levels is observed in the co-occurrence matrix as shown in figure [Figure 5.28](#). This matrix will have most counts along the leading diagonal for smooth textures and more counts in the opposite corners for sharp textures. By looking at the specific pattern of counts, then the texture of many different surfaces can be distinguished. This entire process is repeated a second time, but considering pixel pairs in a vertical orientation to capture texture descriptors in both directions.

To reduce the number of values that must be passed into the learning algorithm several calculations are performed on this matrix to produce characteristic values for the texture:

$$\text{Contrast} = \sum_{i,j} |i - j|^2 p(i, j) \quad (5.9)$$

$$\text{Correlation} = \sum_{i,j} \frac{(i - \mu_i)(j - \mu_j)p(i, j)}{\sigma_i \sigma_j} \quad (5.10)$$

$$\text{Energy} = \sum_{i,j} p(i, j)^2 \quad (5.11)$$

$$\text{Homogeneity} = \sum_{i,j} \frac{p(i, j)}{1 + |i - j|} \quad (5.12)$$

Where i, j are the rows and columns of p which is a normalised version of the co-occurrence matrix (indicating the probability of occurrence for each pair), μ_i is the mean row index, μ_j is the mean column index, σ_i is the standard deviation of row indexes and σ_j is the standard deviation of the column indexes. The contrast is a measure of the intensity contrast between a pixel and its neighbour over the whole image, a value of zero indicates a constant image ([Figure 5.26.a](#)). Correlation

is a measure of how correlated a pixel is to its neighbour over the whole image, the value can range from -1 to 1 for a perfectly positively or negatively correlated image, a value of zero would indicate a random image (Figure 5.26.b). Energy (also known as uniformity) is the sum of squared elements in the GLCM giving a measure of the variance, it has a value of zero for a constant image and has a maximum value of one (Figure 5.27.a). Homogeneity is a value that measures the closeness of the distribution of elements in the GLCM to the GLCM diagonal which gives an indication of how strongly textured the image is as mentioned above, it can range from zero to one where one indicates a perfectly diagonal GLCM (Figure 5.27.b). Note illustrations are only of the horizontal pass.

These calculations are performed on both the vertical and horizontal GLCM providing 8 values to be used in the training data.

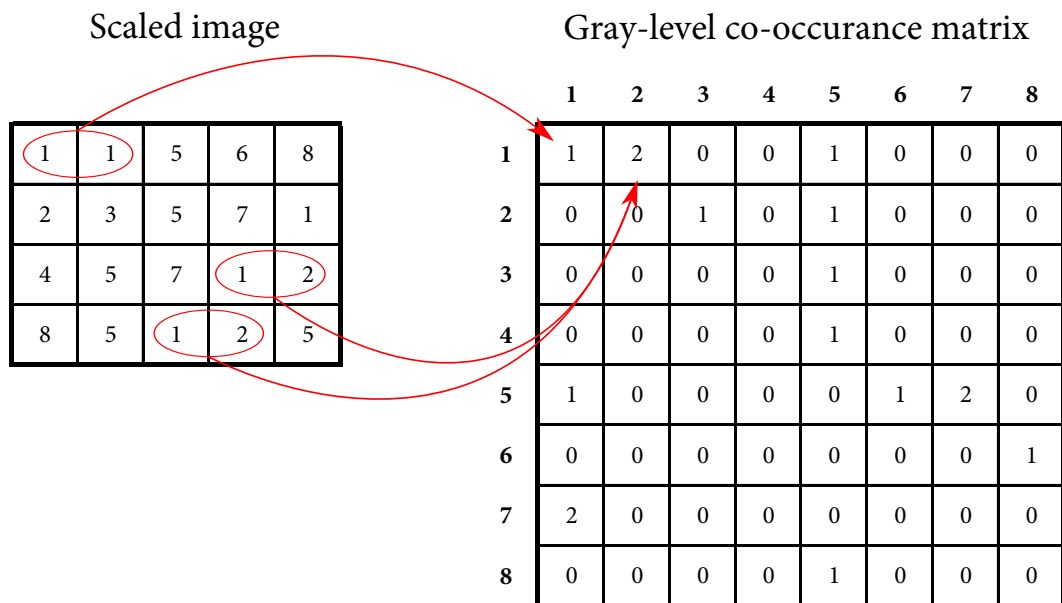


Figure 5.28: GLCM matrix analysis illustration. Recreated from source image found in *Mathworks MATLAB 2010a* documentation.

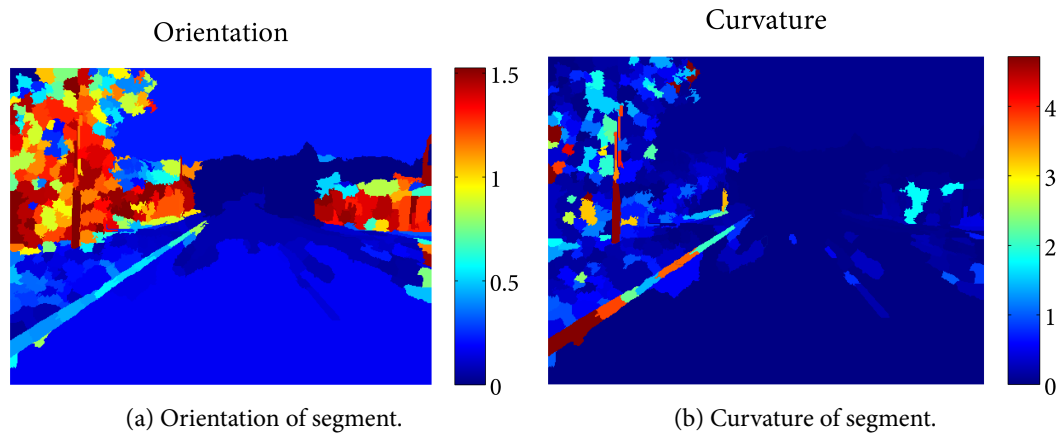


Figure 5.29: Colour coded magnitude maps showing orientation and curvature.

5.5 CHOICE OF DEPTH AUGMENTED DESCRIPTORS

The dense depth maps, produced by the stereo vision software, are converted into 3D point clouds and provide additional properties of the discovered segments. By overlaying the segment boundaries obtained from the fusion segmentation onto the depth map the point cloud for each region can be extracted. Now instead of inspecting image data the point clouds relating to each segment will be analysed to get the physical shape information. See [Figure 5.29](#) and [Figure 5.35](#) for a demonstration, the source data can be seen in [Figure 5.19](#) on page 128.

This section will use the point cloud extracted from a road segment for illustration, this segment and matching disparity image can be seen in [Figure 5.30](#). This process is performed on each and every segment where at least 10 disparity values can be converted to valid 3D co-ordinates. The road segments are generally the largest and most easily visualised, making it a good example for this discussion. Due to the high resolution of the disparity map (512 by 384 pixels) large regions can produce point clouds consisting of over 10,000 points. Performing regression on such a large matrix is computationally expensive and having such an abundance of points is not necessary to establish the general shape of the segment. For these reasons, point segments which have point clouds

consisting of over 1000 points, are randomly sub-sampled down to this quantity. This provides a less dense point cloud for more efficient computations.

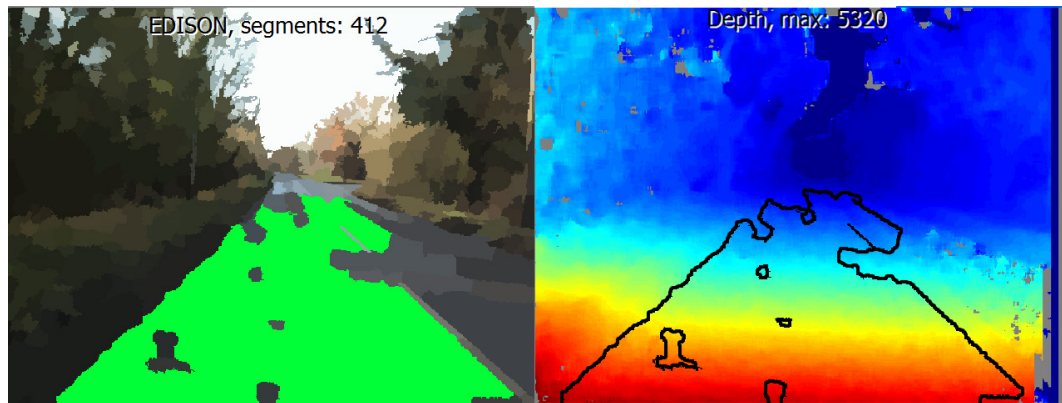


Figure 5.30: Image and disparity map of road segment used to extract point cloud.

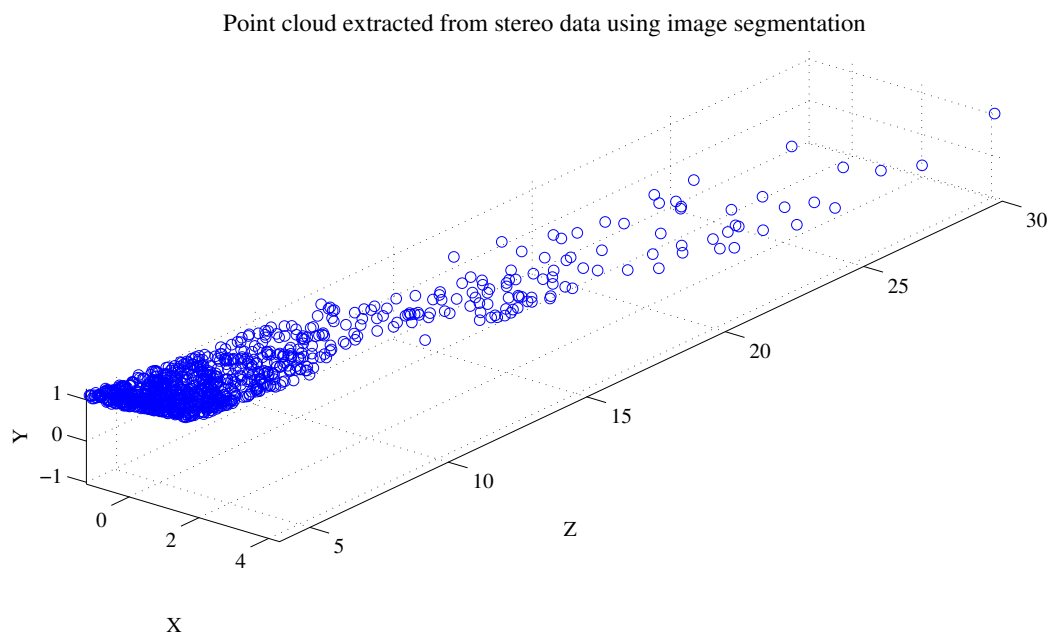


Figure 5.31: Point cloud extracted from road segment. Cloud density reduced to 1000 points using random sampling.

5.5.1 Point cloud orientation

Using the set of extracted 3D points, relating to the segment in question, the first step is to determine its orientation relative to the ground plane. See [Figure 5.31](#) on the previous page for the extracted and sub sampled point cloud for the road segment shown in [Figure 5.30](#). The ground plane in this case is the internally calibrated $y = 0$ plane of the stereo camera co-ordinate system. Orientation is difficult to define in three dimensions, as many representations exist depending on the origin and notation used. Generally a homogeneous rotation matrix, or set of Euler angles, can determine the orientation of one plane relative to another but, without the notion of direction, the same orientation can be written in several permutations.

For example, imagine a wall at the side of the road compared to the road plane, it could be described as a plane rotated 90° about the z axis (distance from vehicle). Equally it could be described as a plane which has been rotated 90° about the x axis followed by 90° about the y axis. Furthermore these angles could equally be expressed as a rotation of 270° or -90° etc. This is a problem if rotations are defined in terms of their angle about the x and z axis as two seemingly identically orientated surface could have opposite orientation descriptors creating data which is not easily classified.

Since the goal of this step is simply to determine if a surface is horizontal, such as the road, vertical, such as a wall, or some value in-between, the orientation about the y axis is unimportant as is the direction of rotation. So instead of recording a separate angle for x and z rotations the two angles will be merged using the Euler norm, that is a single orientation value will be recorded given by $\sqrt{\delta_x^2 + \delta_z^2}$ where δ_x, δ_z are rotations about the x and z axis. The angles will also be normalised to the range $[0^\circ, 90^\circ]$ before this takes place. This value will always be near zero for horizontal surfaces and near $\frac{\pi}{\sqrt{2}}$ for vertical surfaces regardless of their rotation in the y axis.

To determine the orientation Singular Value Decomposition (SVD) [65] is used to find the plane normal. Expressed in the standard plane equation [127]:

$$\theta_x x' + \theta_y y' + \theta_z z' + d = 0 \quad (5.13)$$

The normal is the vector $(\theta_x, \theta_y, \theta_z)$, d is the plane offset and (x', y', z') can be any point lying on the plane. The normal is found using a SVD of the form [65]:

$$\mathbf{M} = \mathbf{U}\mathbf{S}\mathbf{V} \quad (5.14)$$

where \mathbf{U} is an unused unitary matrix, \mathbf{S} is a diagonal matrix of singular values and the columns of \mathbf{V} contain the right-singular vectors of \mathbf{M} which in this case will be possible solutions for the values $(\theta_x, \theta_y, \theta_z, d)$. \mathbf{M} is an n by 4 matrix of point cloud data with an additional column of ones to allow for the offset of the form:

$$\mathbf{M} = \begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & z_n & 1 \end{bmatrix} \quad (5.15)$$

Where n is the number of points in the cloud and the columns containing (x, y, z) are the co-ordinates of the points in the cloud. The best set of $(\theta_x, \theta_y, \theta_z, d)$ can be found in the column of \mathbf{V} with the smallest corresponding singular value. That is, if the i^{th} diagonal of \mathbf{S} is the minimum of \mathbf{S} then the i^{th} column \mathbf{V} represents the values for $(\theta_x, \theta_y, \theta_z, d)$. See Figure 5.32 on the following page for an illustration of the fitted plane.

After obtaining the normal vector of the plane which fits the point cloud the relative angles can be found by producing the appropriate rotation matrix between $(\theta_x, \theta_y, \theta_z)$ and the normal of the ground plane, $(0, 1, 0)$. Assuming \mathbf{R} is such a rotation matrix

Plane of best fit on point cloud, normal is (-0.04, 1.00, 0.00)

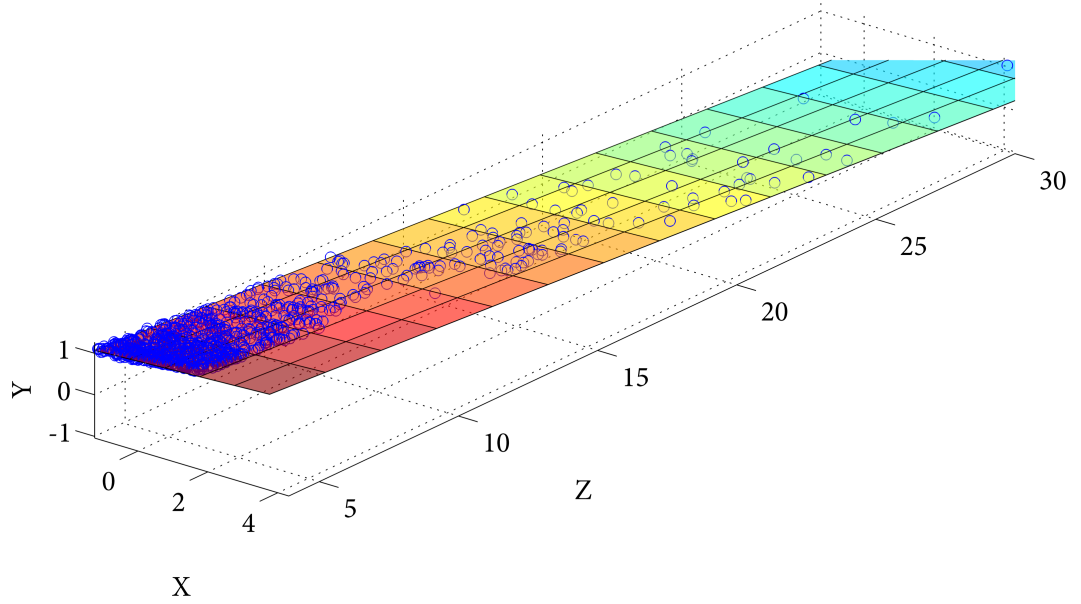


Figure 5.32: Linear surface fitted to road point cloud.

obtained via basic geometry identities then the rotation in the x and z axis will be given by:

$$\begin{aligned}\delta_x &= \text{atan}(r_{32}/r_{33}) \\ \delta_z &= \text{atan}(r_{21}/r_{11}) \\ \hat{\theta} &= \sqrt{\delta_x^2 + \delta_z^2}\end{aligned}\tag{5.16}$$

where r_{ij} is the element at row i , column j in rotation matrix \mathbf{R} , δ_x, δ_z are the rotations about the x and z axis in radians and $\hat{\theta}$ is the final orientation descriptor used in the segment description vector. See [Figure 5.29](#) (a) for an illustration, notice how it cleanly distinguishes between horizontal and vertical surfaces and shows the edge of the curb as being some value in-between. The code listing for this procedure can be seen in [Listing A.3](#) on page [232](#).

5.5.2 Point cloud planarity

The point cloud is a rich source of information. Orientation creates a basic distinction between horizontal and vertical surfaces but it does not indicate the shape of that surface. A row of bushes and a flat wall would be indistinguishable if this was the only measurement. Given such a dense point cloud it should be possible to estimate the extent of curvature present in the surface, or lack thereof. While there are many types of surface that are naturally curved, or warped, in the environment, and each of them will have a particular characteristic shape, it is not important to create a true recreation of the surface, only to determine if some kind of non-linear shape is present. For this reason the most simple non-linear model was chosen; a second order polynomial surface. This model is computationally cheap and the extent of curvature is expressed simply by the squared term coefficients. The model used for this surface will be:

$$y = \theta_1 x + \theta_2 x^2 + \theta_3 z + \theta_4 z^2 + \theta_5 \quad (5.17)$$

where $\theta_1 \dots \theta_5$ are the polynomial coefficients and x, y, z can be any point on the surface. Notice how this model defines y as the dependant variable and there is allowance for offset in the x and z directions.

Expressing the surface in terms of x and z works well for horizontal surfaces but will lead to ill-conditioned coefficients when the surface is close to vertical i.e. one of the linear coefficients will approach infinity. Since the orientation is already known, this problem can be easily overcome by rotating the entire point cloud to have it lie on the $y = 0$ plane. Using the rotation matrix \mathbf{R} obtained in the orientation section, the point cloud is rotated such that its linear plane of best fit now coincides with the ground plane. Secondly the point cloud is re-centred by subtracting the mean x and z position from all points in the cloud. [Figure 5.33](#) shows the result of this transformation. Notice how the corrected

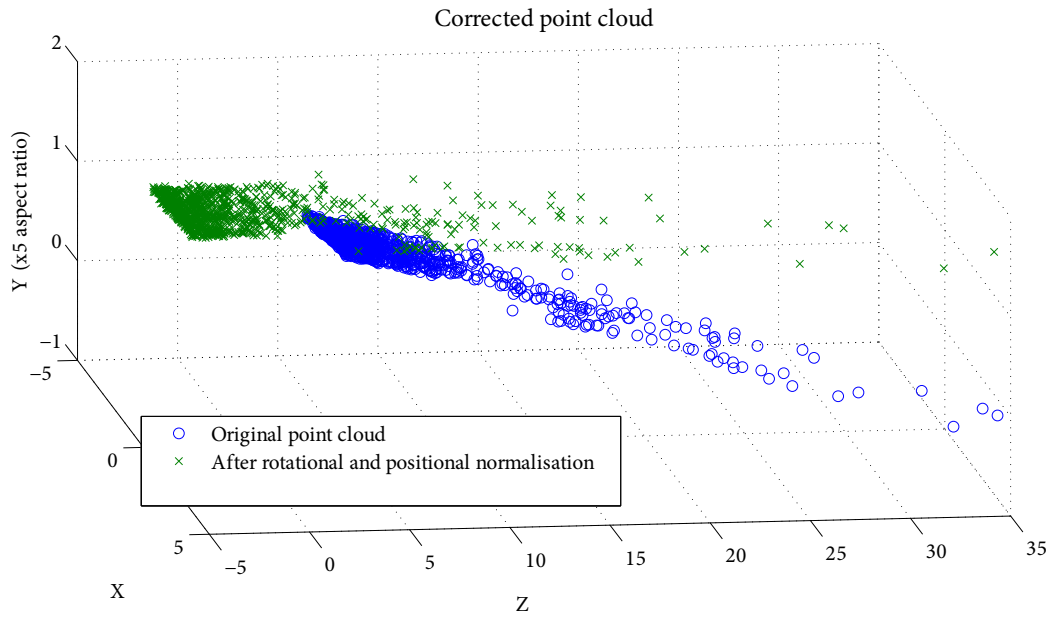


Figure 5.33: Point cloud rotation and offset prior to polynomial surface fitting.

cloud now lies perfectly perpendicular to the ground plane and the most dense part is centred about $x = 0, z = 0$.

With the point cloud now in a suitable condition for plane fitting, the non-linear regression can begin. First the equation is re-written in the form:

$$Y = X\theta + \epsilon$$

where vector \mathbf{Y} contains the height of the points, matrix \mathbf{X} contains the x and z positions relative to the vehicle and a column of ones for the offset, $\boldsymbol{\theta}$ is the coefficient vector and $\boldsymbol{\varepsilon}$ is the noise term.

$$\mathbf{Y} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} x_1 & z_1 & 1 \\ x_2 & z_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n & z_n & 1 \end{pmatrix}, \quad \boldsymbol{\theta} = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \end{pmatrix}, \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix} \quad (5.18)$$

Where n is the number of points in the cloud. In order to estimate $\boldsymbol{\theta}$ robust regression is used to reduce the adverse effects of noise and outliers. There are a number of robust regression techniques in scientific computing papers such as those written by Street et al. [166], Holland and Welsch [78], Dumouchel and O'Brien [39] however it was decided to use an iteratively re-weighted least squares method with a bi-square weighting function due to the favourable results shown in the report by Alma [4]. For each r_i , $i = 1, \dots, n$ the associated weight w_i is:

$$w_i = \begin{cases} (1 - r_i^2)^2 & \text{if } |r_i| < 1 \\ 0 & \text{if } |r_i| \geq 1 \end{cases} \quad (5.19)$$

where:

$$r_i = \frac{\mathbf{d}}{\lambda s \sqrt{1 - \mathbf{h}}} \quad (5.20)$$

and \mathbf{d} is a vector of residuals from the previous iteration of length i , \mathbf{h} is a vector of leverage values from a least-squares fit and s is an estimate of the standard deviation of the error term given by $\frac{\mathbf{M}}{0.6745}$. Here \mathbf{M} is the median absolute deviation of the residuals from their median. The constant 0.6745 makes the estimate unbiased for a normal distribution.

The smallest three absolute deviations are excluded when computing the median. Finally, λ is the tuning constant, a value of 4.685 was chosen to give coefficient estimates that are approximately 95% as statistically efficient [132] as the ordinary least-squares estimates, provided the response has a normal distribution with no outliers. Decreasing the tuning constant increases the down-weight assigned to large residuals; increasing the tuning constant decreases the down-weight assigned to large residuals. See Figure 5.34 for the fitted surface, notice how it shows the road sloping gently across the x direction to allow rain water to run off the road.

After the coefficients are found the only values of interest are θ_2 and θ_4 which are the non-linear coefficients. If these two coefficients are zero then the plane will be perfectly flat, larger values will indicate increasingly pronounced curvature. This measure is useful for separating surfaces that are generally flat such as the road and pavements from surfaces that will follow smooth curves such as vehicles. However, for the same reasons discussed in the orientation section, these two values are combined into one to remove the ambiguity which can occur when referring to the x and z axis. Therefore the final plane curvature value is given by $\sqrt{\theta_2^2 + \theta_4^2}$. Figure 5.29 (b) shows how this measurement maps onto the sample scene, it shows that the majority of surfaces in this scene are not curved but the few that are not such as the edges of the pavement can be easily distinguished.

PHYSICAL TEXTURE Since a surface of best fit has been found, the deviation of the points from the surface can now be examined. The standard measure of this deviation is the **MSE** (mean squared error) which is the average of the squared distance from each point to the fitted surface. The **MSE** represents the amount of deviation that exists between the point cloud and the model, since the model surface is perfectly smooth this value can reveal how scattered or smooth the point cloud is in comparison. Some authors such as Posner et al. [148] use the **MSE** test on surfaces but the best fit model is a flat plane. This approach will always show a high degree of error for surfaces that are naturally curved even if they are smooth

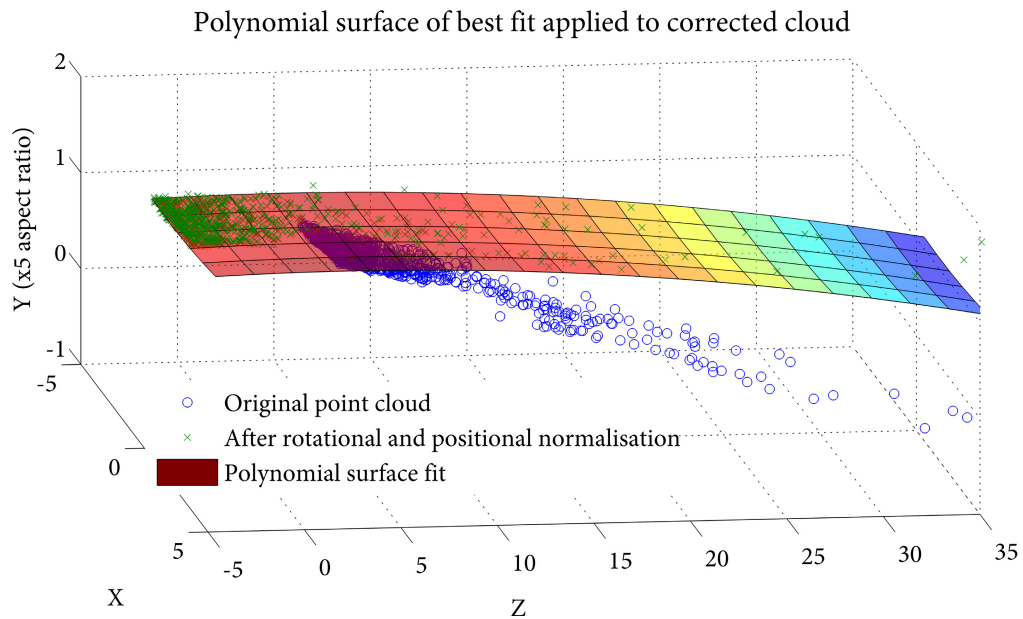
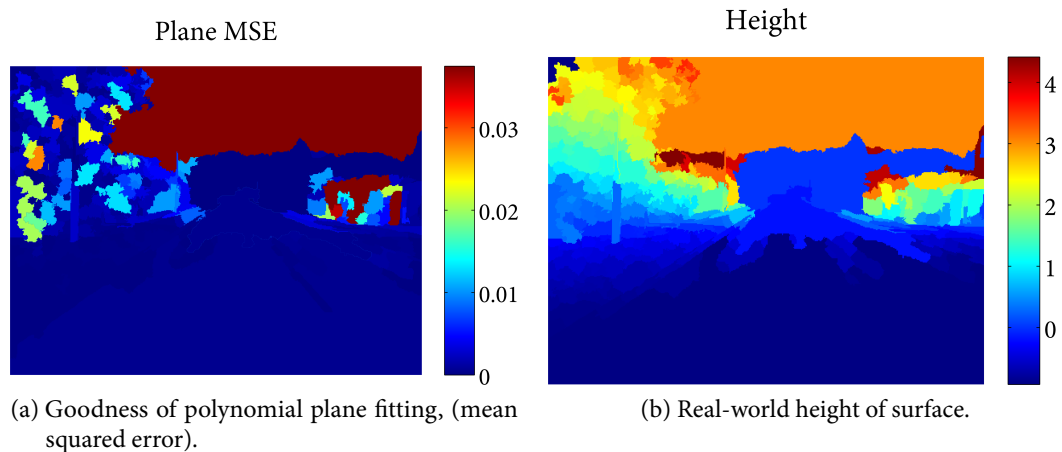


Figure 5.34: Polynomial surface fitted to road point cloud.

Figure 5.35: Colour coded magnitude maps showing height and plane fit [MSE](#).

since a plane cannot accurately represent such a surface. This will create misleading learning data due to lack of flexibility in the model.

Using a polynomial surface means that surfaces that are either flat and smooth, or warped and smooth, will both have minimal [MSE](#). This, in combination with the measure of curvature, allows the identification of surfaces such as curved wall

sections and car body panels characterised by non-zero curvature but low **MSE**. Because curvature has been accounted for and measured the **MSE** will now directly represent the roughness of a surface i.e how scattered it is compared to a smooth continuous surface. For example, surfaces such as hedges will show a different characteristic, they generally exhibit a gentle curvature but the surface is rough and scattered giving a non-zero curvature and a high **MSE**. Since the accuracy of the stereo camera falls off sharply with distance this measure is only applicable to surfaces within approximately 30m of the vehicle. [Figure 5.35](#) (a) visualises the results from **MSE** testing, notice that surfaces that are curved such as the pavement edging (as shown in [Figure 5.29.b](#)) do not necessarily have a high **MSE** as they would if a simple planar model was used for fitting.

PHYSICAL HEIGHT This measure supports the vertical image position measure discussed in the previous section. For segments where a point cloud is available it provides a more robust method of determining a surfaces height compared to others. When looking at vertical image position the distance from the camera is not taken into account, therefore near objects which overhang the vehicle will have maximum vertical position but in reality are closer to the ground than distant, taller objects. The physical height is simply the mean y co-ordinate of all points in the point cloud see [Figure 5.35](#) (b) for a visualisation.

Before moving on to the discriminatory analysis, here is a brief note on the use of robust non-linear regression vs. non-robust non-linear regression. [Figure 5.36](#) and [Figure 5.37](#) show the results of a short test comparing the performance of the non-robust Levenberg-Marquardt [124] algorithm and the robust method described in the planarity section above. The point cloud pictured is a road section similar to the one used in the previous examples but not the same. It is immediately obvious, just by comparing the **MSE** result (< 0.001 for robust and approximately 0.03 for non-robust), that the robust method outperforms the non-robust method by an order of magnitude. [Figure 5.37](#) clearly shows

the fit is being heavily influenced by the small group of outliers in the lower right causing the surface to peel away from the dense group of points in the centre.

However, although the robust method is clearly superior, the drawback is that the computation time is, on average, 80% longer than the non-robust method. The robust method can take up to one second on particularly heavily segmented images which would make it unsuitable for real-time application on present hardware without further optimisation, or parallel computation (which would be easily implemented since each segment analysis in an image is independent of each other). Furthermore, informal testing shows that using the non-robust method does not significantly impact on classifier training performance since the curvature and MSE measures have similar distributions for both methods. Therefore the choice between the two will depend on the application and goals of a particular project, in this case the aim is simply to research possible methods so the robust method is used.

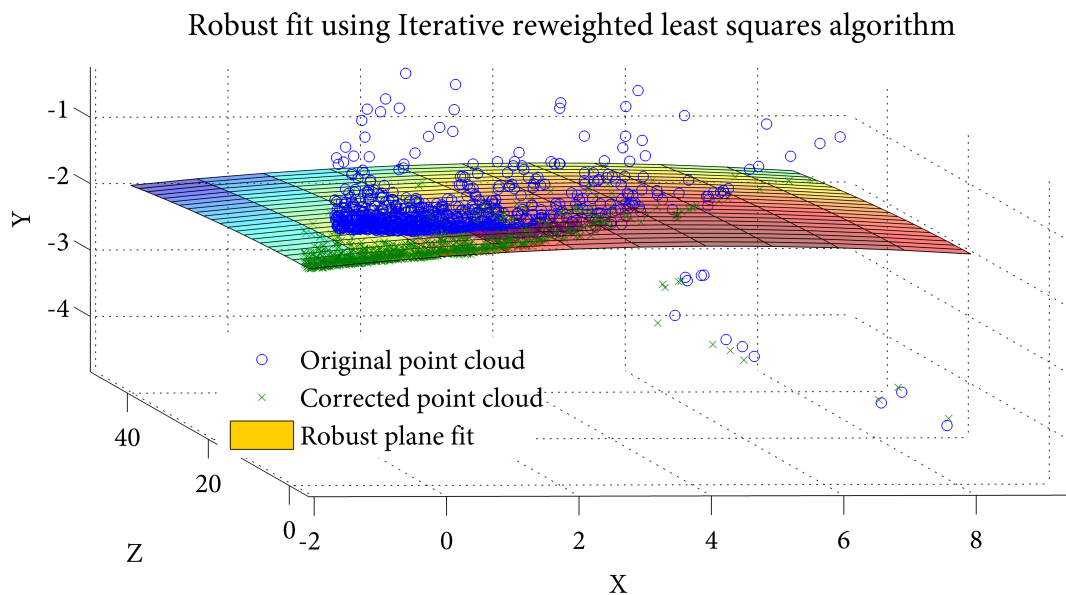


Figure 5.36: Example of robust plane fitting, MSE is less than 0.001.

Non-Robust fit using Levenberg-Marquardt nonlinear least squares algorithm

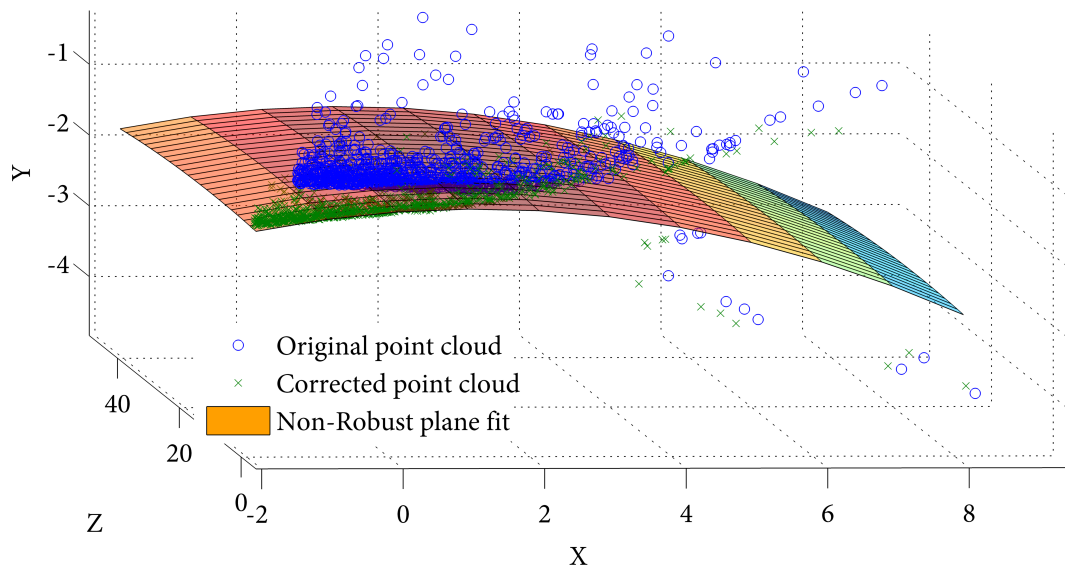


Figure 5.37: Example of non-robust plane fitting, MSE is 0.03. Implemented as described in papers by Seber and Wild [160], Seber [161].

5.6 DISCRIMINATORY ANALYSIS WITH PRINCIPAL COMPONENT ANALYSIS

In order to determine the significance, or amount of unique information provided by each of the features measured above, a [PCA](#) was performed. Ideally there will be minimal correlation between any pair of measures in the set which would mean that each feature is contributing discriminating information about the segment being measured. If any measurements are strongly correlated, then their information is redundant, because knowing one measurement essentially tells you what the second measurement will be. In this case one of the features can be eliminated. It is important to remove redundant features because, if left in, then the information described by the feature will carry double the weight in training, since the classifier will see two inputs showing the same pattern. It also slows down the training process and makes all steps of data analysis less computationally efficient.

The [PCA](#) provides an easy to visualise method of identifying measurements that are strongly correlated. To perform the analysis 20 images were selected from the “Berkswell Manual” dataset (samples can be seen in [Section 4.3.4](#) on page 98). Each of these images was segmented as described and each segment was analysed to produce the description vector consisting of the 25 descriptors. These description vectors were collected into a data matrix with 25 columns, one for each descriptor, and approximately 6000 rows, one for each segment. The goal was to find any connection or relationship between the columns.

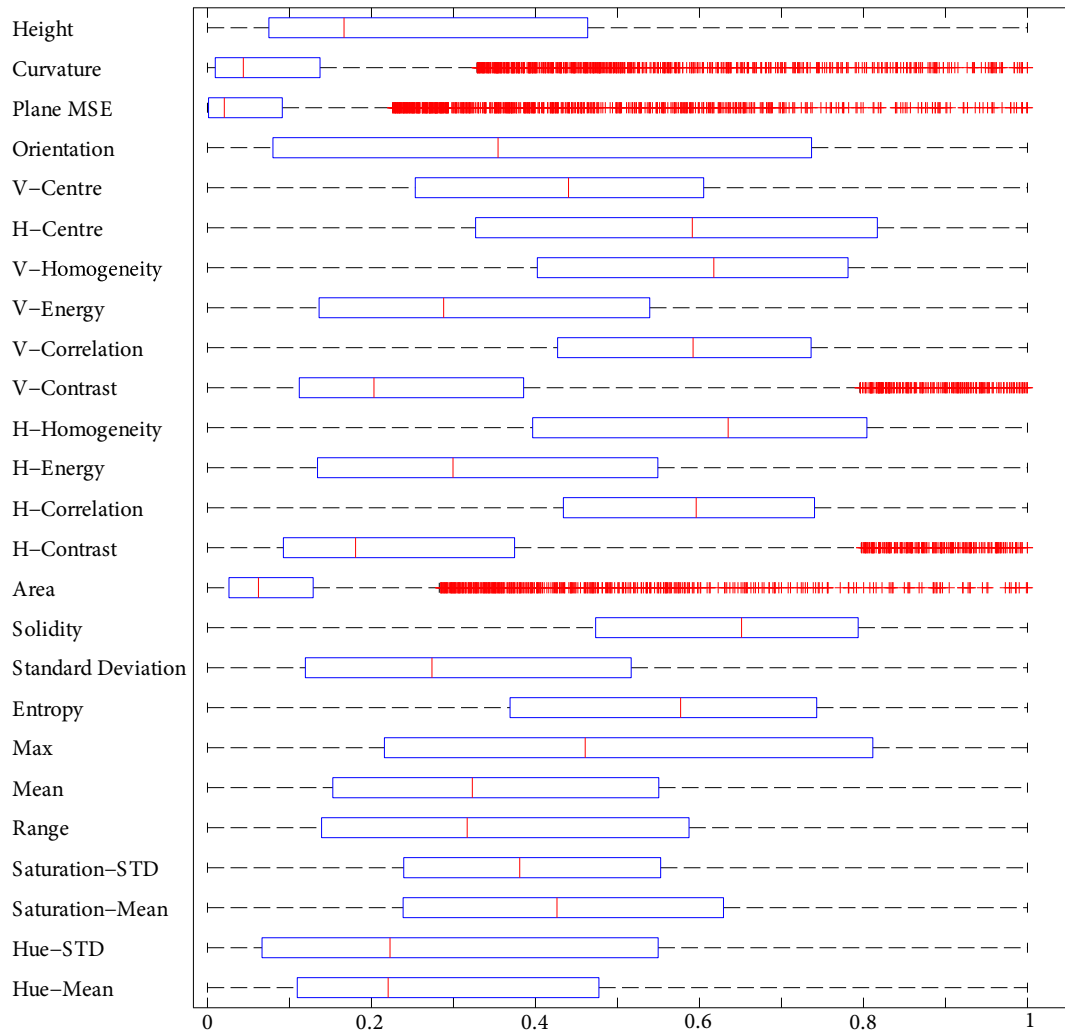


Figure 5.38: Box plot of segment measurement data after scaling each set independently to the range $[0, 1]$. The central red mark is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to cover all points which satisfy the inequality $q_1 - 1.5(q_3 - q_1) < x < q_3 - 1.5(q_3 - q_1)$ where q_1 and q_3 are the 25th and 75th percentiles respectively, x is a sample point and the value of 1.5 is the default separation factor, refer to Velleman and Hoaglin [181]. The red ticks are samples falling outside this range and represent points within large leading, or trailing tails, on the distribution. The presence of these ticks would indicate a skewed distribution.

This data exists in 25 dimensions, where each of the descriptors is an axis label. The distribution of data points along each of these axes explains some percentage of the total variation in the dataset across all dimensions. The idea of [PCA](#) is to transform this 25

dimensional space onto a new space where each axis is chosen in such a way that it explains as much of the variance as possible. The benefit of this is it makes visualising the data possible on a standard 2D plot. By plotting the data on the first two principal components, the majority of variance in the set can be visualised along with how each of the original axes maps to the principal components.

First look at [Figure 5.38](#), it provides a useful summary of each descriptor discussed above and shows some basic information about the distribution of each set. The caption describes that the presence of red ticks indicate a heavily skewed distribution which can be seen in the measurements of curvature, [MSE](#), vertical and horizontal [GLCM](#) contrast and area. The skewed distribution in the curvature and [MSE](#) are easily explained since the column contains many instances of missing data. Missing data in these columns occurs whenever there is insufficient depth information for a segment, either due to excessive distance, or missing disparity data. When this occurs zero is used for these measurements. In addition, even when these measurements are not missing, the calculated value is generally near zero causing any large values to sit outside the expected zone. Velleman and Hoaglin [[181](#)] refer to these points as outliers however the term does not apply here due to the large quantity of samples in the upper region and the fact that this is not necessarily a normal distribution. The vertical and horizontal [GLCM](#) contrast show a skewed distribution because, in general, a segment will only contain a single type of surface and consequently the variation, or contrast, across the segment itself will be minimal. However on the occasion when the segmentation fails to separate different surfaces the contrast measurement will respond to the presence of two, or more, different intensity levels causing uncommonly high values in this column. Finally the area column shows this behaviour because, as previously discussed, the vast majority of segments will be at a minimum size dictated by the mean-shift step of the segmentation. The large segments, lying outside the normal range, will be exclusively sky, road and pavement segments.

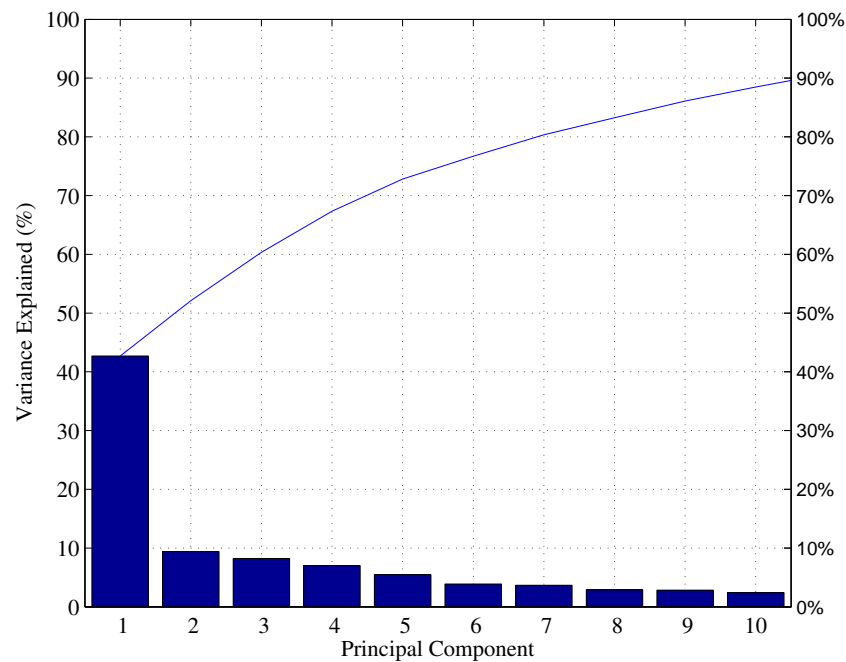


Figure 5.39: Scree plot, variance explained by each principal component.

After normalising each column in the data matrix to the range $[0, 1]$ and performing the PCA Figure 5.39 is obtained. It shows that using the first three principal components approximately 60% of the variance can be explained. If the data points are plotted against these axis it should give an accurate illustration of which features are contributing most to the variance and therefore are the most suitable for separating one class from another. The additional variance explained by the additional components, after the first, does not drop off steeply indicating that the information cannot be easily reduced in dimension and therefore the existence of unique information in most if not all columns.

Figure 5.40 shows the observations plotted against the first and second principal components. The original axes have been transformed and plotted on the graph too. Original axes that are mostly horizontal (i.e. closely aligned with the 1st principal) are the strongest contributors to the overall variance of the cloud, it clearly shows the range, height, standard deviation and GLCM contrast values contribute a large weighting to the 1st principal component, which itself accounts for 42% of the overall variance.

More importantly it shows that the vertical and horizontal components of the respective **GLCM** measurements are strongly correlated i.e. they share almost identical vectors. The overlapping of the vertical and horizontal measures would suggest that the majority of textures being dealt with here are rotationally invariant i.e. the pattern looks the same regardless of which direction you inspect it in. Although textures are rotationally invariant in most cases, the cases where it is not will provide a useful form of discrimination between particular surfaces that have this property. This strong correlation can also be observed in [Figure 5.38](#) on page 152, the box plots for the respective elements of the vertical and horizontal **GLCM** features are strongly similar in terms of mean and quartile distributions.

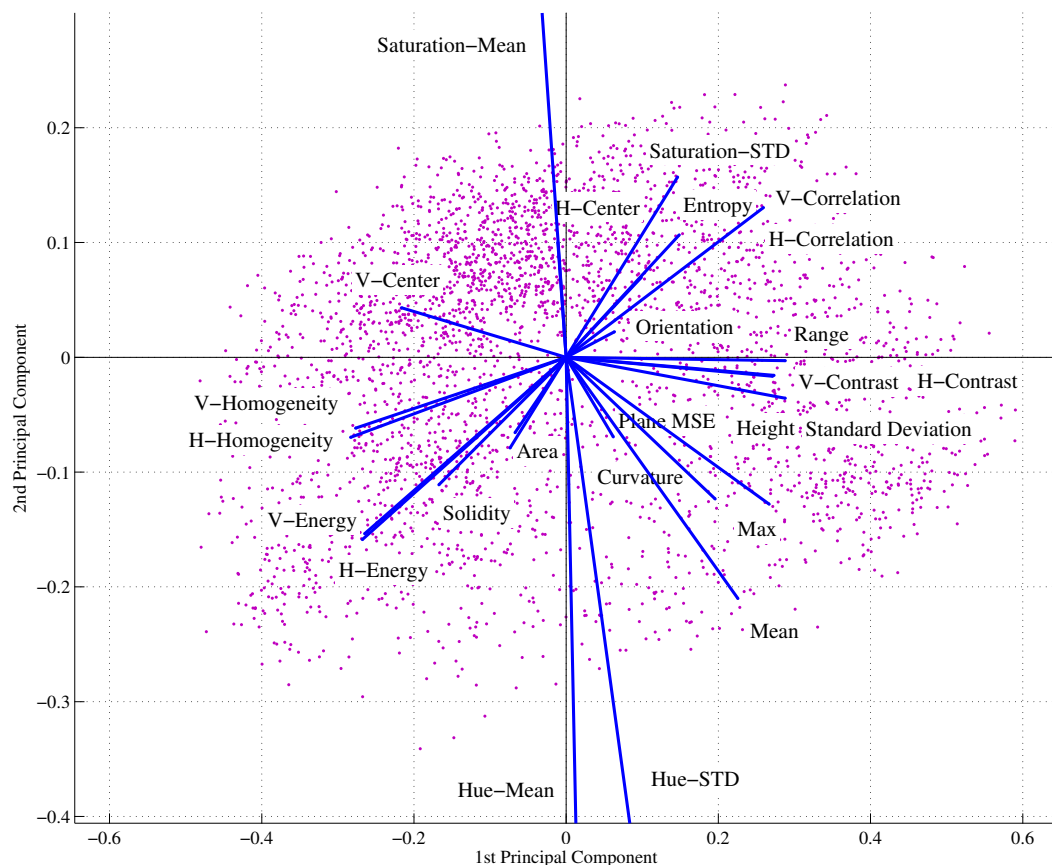


Figure 5.40: Loading plot, 1st and 2nd principal components showing close relationship between vertical and horizontal **GLCM** features.

To verify if the [GLCM](#) measures are indeed nearly 100% correlated [Figure 5.41](#) shows the 1st and 3rd components again with the same data, and original dimensions of the data, transformed onto these axes. From this perspective, a significant amount of deviation can be seen between the respective vertical and horizontal components, particularly in the homogeneity and the correlation factors. Considering the two graphs together, it is clear that each feature in the description vector is contributing a significant, and necessary, portion of the variance and therefore there is no need to remove any one of them.

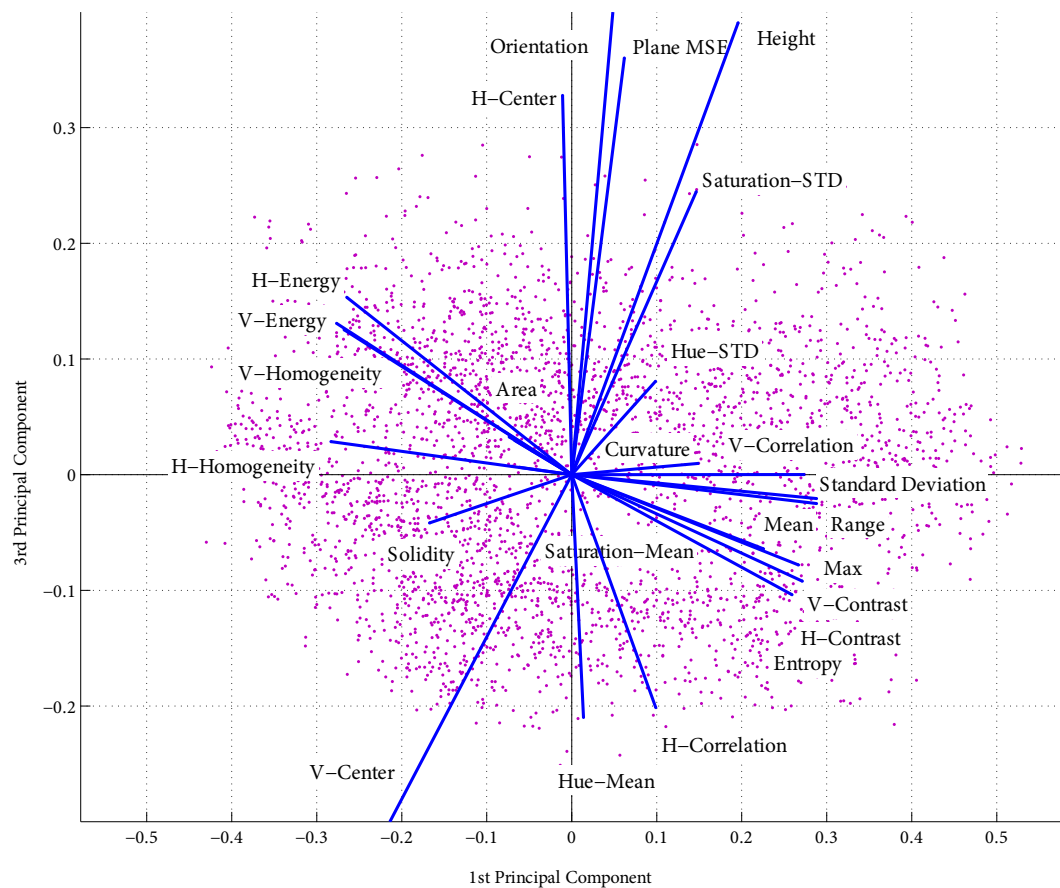


Figure 5.41: Loading plot, 1st and 3rd principal components.

5.7 CONCLUSION

In this section the most important decisions regarding the image processing pipeline have been explained, reviewed and justified, looking at [Figure 1.6](#) on page 16 all the image, and depth map, processing tasks between raw data input and classifier training have been discussed here.

To summarise, images are first colour corrected to reduce variance under different lighting conditions then segmented using a novel combination of edge, mean-shift and depth segmentation. The use of edge segmentation preserves fine separations between similarly textured regions, while the mean-shift prevents over segmentation of surfaces with strong textures. It was found that depth segmentation provides edges that must always be present in the segmentation, but often these edges are already identified in the edge segmentation step making it an optional addition.

After segment boundaries are established the image is converted into secondary forms to suit various texture measurements. One form is a [HSV](#) colour space representation to provide lighting invariant colour and saturation information. Another is a grey scale version to provide texture information via [GLCM](#) metrics. After texture descriptors have been collected, the segmentation map is applied to the depth map providing a localised 3D point cloud for each identified segment. Using this point cloud orientation, curvature, surface roughness and physical height are determined using a polynomial surface fitting technique, which is more suitable for natural surfaces than existing approaches using flat planes.

Finally the proposed segment measurements were performed on a batch of real images and the dataset was analysed to verify that each descriptor was necessary and providing a significant degree of discrimination. This must be established in order to get good performance from segment classifiers which will be introduced in the next section. The

next section will use the descriptors described here to train two types of classifier and then use them to label road images. The Support Vector Machine ([SVM](#)) and neural network classification methods will be used and compared using identical sets of data.

6

SUPERVISED LEARNING WITH MANUALLY LABELLED IMAGES

CONTENTS

6.1	Data preparation	160
6.1.1	Frame selection and manual labelling	160
6.1.2	Labelled dataset description	164
6.2	Experiment setup	168
6.2.1	Support vector machine configuration	169
6.2.2	Neural network configuration	172
6.2.3	Comparison methodology	176
6.3	Training results	177
6.4	Conclusion	187

In this chapter the two most commonly used classification methods, neural networks and Support Vector Machine (SVM)s will be compared using the same training data. The aim is to make the test as fair as possible by configuring each classifier for best performance using the advice from other authors in the field. While there have been several studies comparing the two classifiers on land use classification from geographical surveys [70, 191, 164] there is not yet a comparison relating to mixed visual and spacial training data. This comparison is intended to act as a guide for future research teams when deciding which classifier to select.

The first part of this chapter will discuss how training data was obtained and how the dataset was divided for training and testing purposes. Next the rational used to configure and tune the two algorithms is presented. Finally the a comparison of training results is given after using the two algorithms to categorise image segments into one of 10 target classes. These classes are “Road”, “Sky”, “Pavement”, “Grass”, “Road Object”, “Road Marking”, “Wall”, “Vegetation”, “Pavement Object” and “None” (see [Figure 6.3](#) on page 166 for more detail).

6.1 DATA PREPARATION

6.1.1 *Frame selection and manual labelling*

To train any supervised machine learning algorithm such as [SVM](#) or neural networks examples of correctly classified data are needed. In general, the more training data available, the more successful the training will be. It is also important to ensure that the diversity of the training samples is as great as possible, i.e. they are distributed across the entire spectrum of different cases that may feasibly be encountered in the testing data (and later in the real world).

While it is ideal to have as much data as possible, practical limitations, such as training time and time taken to label the correct classes, means that a compromise must be made. In general, work detailed in papers on machine learning in similar areas such as [\[42, 148\]](#) state that training sets consisting of at least 8000 samples have been used. In addition to the training data, labelled samples are also needed to verify the performance of the classifier after training. This set must be kept separate from the training data. Based on this, it was decided to aim for a dataset of 16, 000 labelled segments to provide sufficient samples for both the training and testing sets.

Since, on average, each video frame is split into 400 segments that would mean that 40 labelled images would be needed to meet this target. In order to ensure the training data is sufficiently *general* i.e. the training samples represent road scenery from a broad range of locations and conditions, a selection of frames were chosen from the two most diverse datasets – the “Berkswell Manual” set (see [Section 4.3.4](#)) and the “RX Residential set” (see [Section 4.3.2](#)). If the frames were taken from only a single set then there would be a risk that the classifier would only be able to perform well on a certain type of scenery, for example, motorways and have poor performance in any other environment. It is also important that consecutive frames are not chosen since frames captured $\frac{1}{15}$ s apart will show an almost identical scene and contribute no diversity to the dataset. It is partly for this reason that the “A46” was excluded from the training data. When driving on a dual carriageway there is very little variation in the scenes observed. Also, due to the speed of the vehicle (causing motion blur), almost every frame in the set suffered from heavy disparity map noise. Including data with such distorted depth and shape data would pollute the training data.

With these considerations in mind 20 frames were chosen from each included data set. Each frame was checked to ensure the associated depth map was of acceptable quality (i.e. no large regions are missing or filled with noise). The frames were chosen such that they are separated by at least one second (15 frames).

To get the correct target classification for each segment, in each image in the chosen set, manual labelling was used. [Figure 6.1](#) shows the software which was developed for this purpose. Each image is tagged using the correct class labels identified by the corresponding “paint” colour. The labelling happens at the pixel level i.e. the images are not segmented prior to this task and the painted labels can be applied without any constraint. This was done so that the labelling is independent of the chosen segmentation method, allowing different strategies to be compared if other researchers wish to use the data. Once a segmentation is applied to the image, the label image created here is used to

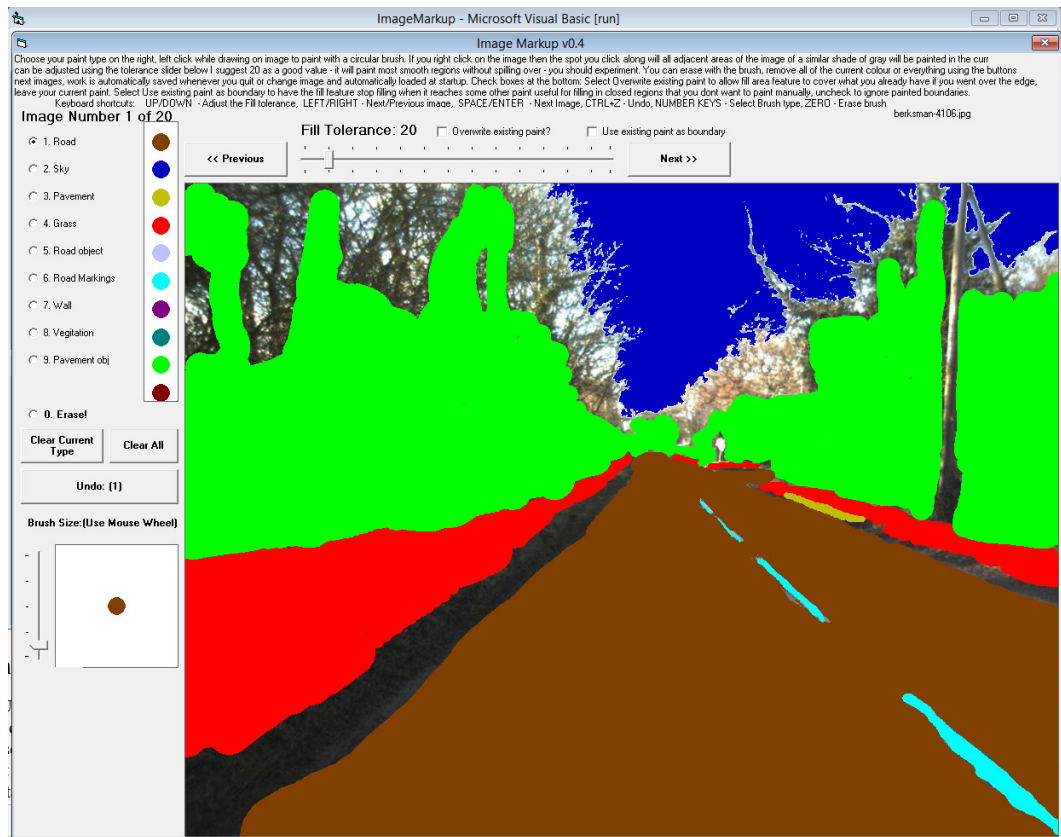


Figure 6.1: Software used for manual training, each frame is tagged using appropriate label using a “brush” at the pixel level. These tags provide learning targets for the classifiers.

derive the correct class for each segment. This is done by applying the same segmentation to the label image, the correct label is then chosen to be the label type covering the majority of the pixels in each segment. This means that if the borders between label types are different in the automatic segmentation, and the label image, each segment will still be assigned the correct class using this majority rules scheme. An example of this is shown in Figure 6.2, notice how each segment in the automatic segmentation takes the class of the most common pixel class within it.

The paradigm for tagging the images was to only label parts of the image that unambiguously represent the description of that class. The fact is that, without defining an excessive number of classes, not all parts of an image will fall exactly into one of the categories defined. A particular example, which is illustrated in Figure 6.1, is the part of

the image where the sky can be seen through the branches of the trees. It might make sense to label these areas as “Vegetation” along with the dense areas of the foliage, however these two types of region have vastly different appearances. The goal of a classifier is to identify the class of a segment based on a particular set of properties that distinguish it from all the other classes. It does this by assuming that members of each class will have similar properties. Therefore if areas of foliage, interspersed with patches of sky, are grouped together with areas of pure foliage, then this assumption will not be valid and the classification performance will be poor. For example the texture properties discussed in [Section 5.4](#) will have significantly different outcomes for each of the two foliage variants. For the same reasons it is also important that no region, which should be labelled as one of the classes, is left unlabelled. If this happens the unlabelled regions will be learnt as “None” which will create a tendency to incorrectly classify true classes as “None” and increase class confusion. In [Figure 6.2](#) it can be seen that manual pixel labelling near the lamp posts leaves a gap between the “Vegetation” class and the “Pavement Object” class to avoid mixing the class definitions. When the image is segmented, and classes are assigned using the majority pixel class, the gap is not present and the edge between regions is accurately defined by the edge detection segmentation step (see [Section 5.2.1](#)).

It is preferable to leave some of the image untagged, rather than to have regions that only partially fit the class description included which will contaminate the training data. This leads to an important point: all areas in the image that are left untagged will also be considered to be a class. This class named “None” is a garbage class to contain any segment that does not belong in one of the other groups. It is important, when classifying unseen images, to have some notion of what does not belong to any of the trained classes, so that if a new type of surface is encountered it can be identified as “new” and not simply classified as the class it most closely resembles. This is the purpose of the “None” class to attempt to identify any image region which does not have a correct classification. Since the “None” class will contain a wide variety of different surfaces, and samples which are

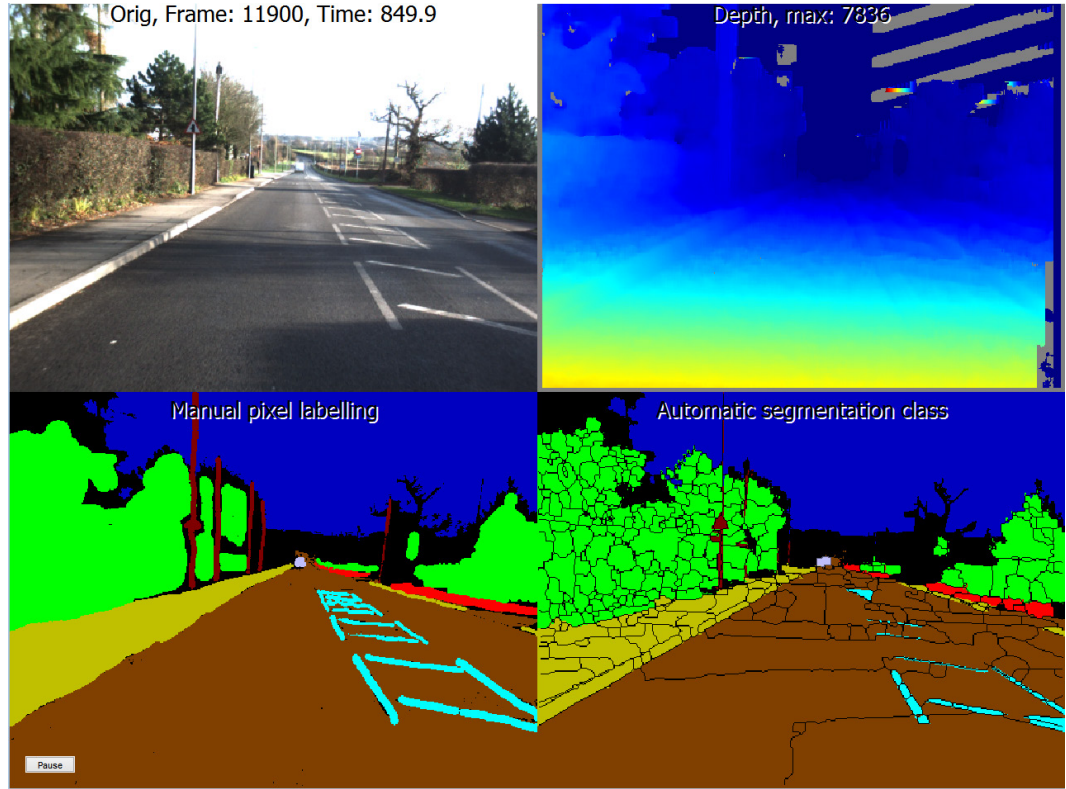


Figure 6.2: Deriving target class from pixel tagging. Top left: colour corrected image, top right: corresponding disparity map, lower left: hand labelled pixel map, lower right: automatic segmentation applied to hand labelled pixel data, most common pixel type is chosen as class for each segment.

unlikely to have any properties in common, the classification performance of this class is expected to be low (for the reasons stated in previous paragraph). However the ability to filter out some regions that should not be classified is preferable to having these segments filled with random, or incorrect, classifications.

6.1.2 Labelled dataset description

After processing all of the 40 images, the total number of manually classified samples was 15,394 which was sufficient to create a large training set and smaller testing set. Let this count be denoted by n_a for the remainder of this chapter. The samples created by analysing each segment are stored in an n_a by 25 matrix \mathbf{D}_a , one column for each of

the features measured (see [Figure 5.38](#) on page 152 for a list of features). The classification targets derived from the hand tagged images are stored in a n_a length vector T_a . At this stage the samples no longer retain any affinity to any particular image, or dataset, this information is not retained during extraction.

[Figure 6.3](#) shows the distribution of class types. It is clear that the classes are not balanced in terms of membership count. In general unbalanced classes can cause problems, because classifier performance is usually measured in terms of *accuracy* which is defined as the number of correct predictions divided by the number of samples predicted. Consider the simple case of a binary classifier which is to be trained with unbalanced classes with 95% positive and 5% negative cases. A training algorithm which is only aiming to maximise the overall *accuracy* could simply predict 100% of samples to be positive and achieve a 95% accuracy rating. While this appears to be a strong performance indicator, the classifier is actually useless as it is scoring 0% accuracy for the minority case. This type of training bias can be avoided by balancing the classes in the training set. Batista et al. [16] presents a study on the problem and a comparison of various solutions. In the case of this research a solution which is applicable to both [SVM](#) and neural network training must be used to keep the comparison fair. The most straightforward approach to balancing the classes, independently of the classifier to be used, is by reducing the number of samples in the larger classes (down or under-sampling) or by increasing the number of samples in the smaller classes via duplication (up or over-sampling). Batista et al. [16] concludes that out of the two methods up-sampling consistently produces better results. This is intuitive since discarding large portions of training data can only reduce the diversity of the training data and adversely affect performance.

To implement this solution a weighting factor is used to give each class an equal sample count and therefore equal training bias. [Table 6.1](#) shows how the weight is calculated to be inversely proportional to the the relative class size. The count ratios for the entire dataset are used for this purpose as this should be representative of any chosen training sub-set.

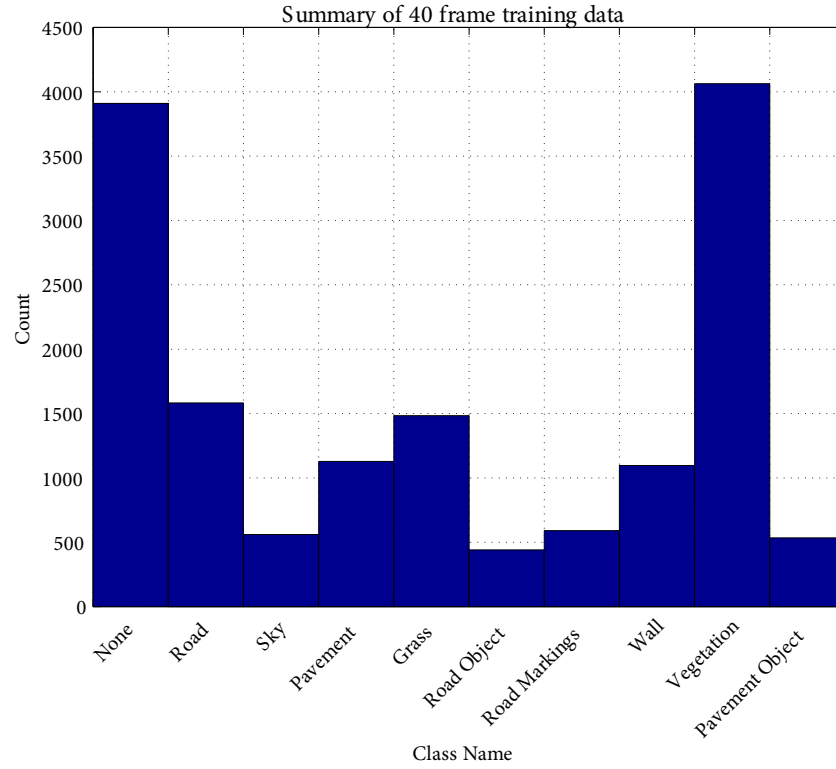


Figure 6.3: Class membership counts to visualise the extent of class imbalance.

$$W_k = \frac{\max(S)}{S_k} \quad (6.1)$$

Where W_k is the weight of class k , S is a vector containing the member count for each class numbered $k = 1 \dots 10$ and S_k is the member count of class k . The largest class, “Vegetation”, is $\max(S)$ so it given a fixed weight of 1.00 then all other classes are up-sampled by the factor shown (weight) such that the sample count matches that of the largest class. When up-sampling classes existing members are duplicated in sequence until the desired count is met.

The list of class types used for this training are listed in [Table 6.1](#). The class “Road Object” is used to mark any type of vehicle encountered on the road, it does not include road furniture which is classed as “Pavement Object”. The “Road Object” is the most infrequent

Table 6.1: Summary of classes and weightings, corresponding histogram can be seen in [Figure 6.3](#).
Total samples, $n_a = 15,394$.

CLASS	SAMPLES	WEIGHT
None	3910	1.05
Road	1582	2.55
Sky	562	7.92
Pavement	1128	3.66
Grass	1483	2.74
Road Object	441	9.05
Road Markings	595	7.08
Wall	1096	3.53
Vegetation	4063	1.00
Pavement Object	534	7.44

since, in general, only a small region of any image will contain a vehicle. Because of the different coloured sections making up a vehicle (e.g. black windscreen, coloured body panels etc.) they generally get over-segmented in both the edge segmentation step and the mean-shift segmentation step (see [Section 5.2.1](#)) causing them to remain over-segmented in the final segmentation fusion. This means that over 10 segments are generally produced for each vehicle encountered in a frame. “Road Markings” is any painted lines on the road this could be useful for detecting pedestrian crossings and chevron lines. “Vegetation” is any tall green plant life including hedges, tree foliage and walls covered in climbing plants. Due to the generally large image coverage, and high segmentation rate, this is the most common class. Finally the “Pavement Object” class includes any object protruding from the pavements including lampposts, road signs and bollards.

Before splitting the dataset into training and testing sets each column was independently normalised to the range $[0, 1]$ as so:

$$N_j = \frac{F_j - \min(F_j)}{\text{range}(F_j)} \quad (6.2)$$

where N_j is the normalised column j in \mathbf{D}_a and F_j is the original column. The subtraction of the minimum and the division by the range are applied element wise to every element in the column. When this is done, the original minimum and range values are recorded for later use. They are needed to scale any new data which is given to the classifier using the same transformation as was used on the training data. If new data is normalised independently it will not necessarily have the same scaling as the training data and this will reduce the performance of the classification. See [Figure 5.38](#) on page 152 for basic distribution statistics on the scaled data.

To split the dataset, \mathbf{D}_a , into a training set and a separate testing set it was decided to use a 80% – 20% split. Thus the training set, \mathbf{D}_r will be a random group of 12,315 samples from \mathbf{D}_a and the training targets, T_r will be the corresponding class labels from T_a of the same length. This leaves the remaining 3,079 samples of \mathbf{D}_a as testing data which will be denoted as \mathbf{D}_s along with the corresponding class labels for verification, T_s . The same training and testing sets will be used to evaluate each of the two classifiers. After splitting the training data \mathbf{D}_r is now up-sampled using [Equation 6.1](#) to give an equal number of samples per class.

6.2 EXPERIMENT SETUP

In order to make a comparison between two systems it is necessary to ensure that both classifiers are operating under optimal conditions, otherwise any conclusions will be meaningless. In order to make the test as fair as possible efforts have been made to adhere to best practices wherever possible. Several studies have been made regarding the optimal ways to configure [SVMs](#) [185, 80] and neural networks [129, 114], these suggestions have been followed where appropriate. This section describes how each of the classifiers were tuned for optimal performance and gives details on comparison methodology.

6.2.1 Support vector machine configuration

First of all it should be noted that, unlike neural networks, SVMs are inherently binary classifiers, samples are divided by a hyperplane and can therefore only fall into one of two categories. Therefore when the term SVM classifier is seen in this thesis, what is actually being referred to is a group of many SVMs working alongside each other to give the behaviour of a multi-class classifier. There are two ways of using a group of SVMs to achieve this; the “one-against-all” [106] method and the “one-against-one” [90, 91] method [31].

Assume a classification problem has k classes; the one-against-all approach will train k classifiers, each one separates samples of one class from all other samples. The one-against-one method trains $k(k - 1)/2$ classifiers, one for each pair of classes. Each classifier is trained only on data belonging to one of the two classes it has been assigned to and its prediction will indicate which of the pair each sample belongs to.

The drawback of the one-against-all approach is that generally more than one of the classifiers will give a positive result, so then the final class of a sample is decided by the positive classification with the largest discriminant (also known as the decision value or distance from the class division hyperplane). However, since the classifiers are independent, the fact that one classifier predicts positive membership with a larger discrimination than another, does not mean it is more certain. The magnitude of the discrimination is not directly comparable across classifiers. The one-against-one approach avoids this problem by using a voting system. The result of each pair-wise classifier will be a vote for one class, then the class with the most votes is the result. In this way the internal discrimination of each classifier does not need to be compared. For this reason, and because the method has been shown to be competitive with other methods by Hsu and Lin [80], the one-against-one method was used. In this experiment there will be 10 output classes, therefore a bank of 45 SVM classifiers will be trained.

The next important decision to be made relates to the choice of kernel function for the SVM (again referring to the entire group which will be configured identically [44]). As discussed in Section 2.3.2 the Radial Basis Function (RBF) kernel has been shown by Hsu et al. [81] to perform at least as well as the linear kernel and generally performs well compared to other choices [80].

One of the key benefits of the RBF is that it only requires a single tuning parameter – the scaling factor, γ . In addition to the box constraint C , which is needed regardless of the kernel choice, this leaves only two parameters to find. Chang and Lin [29], authors of “LIBSVM” the SVM code used for this research, suggest the best way to find these parameters is with a simple grid search. This was done as suggested by methodically testing every pair of C, γ values in a grid pattern, which will be called nodes. At each node the training set D_T was used to train the classifier producing a performance estimate.

To ensure the estimate at each node is a fair comparison the training data is further divided into five groups for five-fold cross validation i.e. the classifier is trained five times at each node, each time using a different set of four out of the five subgroups, and tested on the final subgroup. This ensures each sample is used exactly once in training and testing. The average performance (accuracy) of the five training results is used as the performance estimate.

Figure 6.4 shows the results of the grid search, because the optimal values of these parameters can lie in a range that spans several orders of magnitude a logarithmic search space is used. Initially a coarse search is performed with $\log_2(C)$ values of $(-5, -3, \dots, 13, 15)$ and $\log_2(\gamma)$ values of $(-15, -13, \dots, -1, 1)$. After a peak was found near $(1, -3)$ a high resolution search is performed on the local neighbourhood with steps of 0.25. After completion of the search the optimal pair of parameters was found to be $C = 1.5^2 = 2.828$ and $\gamma = -4^2 = 0.0625$ giving a cross validation accuracy of 85.1%.

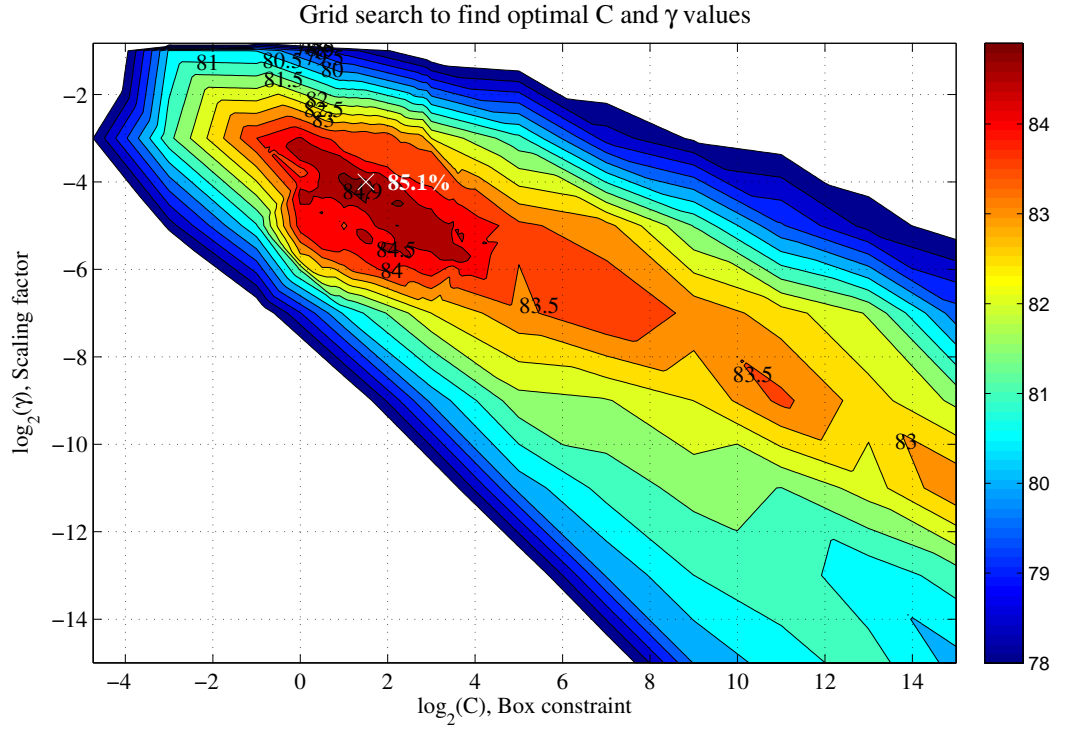


Figure 6.4: Grid search for [SVM](#) parameters, initially a coarse search is performed with $\log_2(C)$ values of $(-5, -3, \dots, 13, 15)$ and $\log_2(\gamma)$ values of $(-15, -13, \dots, -1, 1)$. After a peak is found near $(1, -3)$ a high resolution search is performed on the local neighbourhood with steps of 0.25. After completion of the search the optimal pair of parameters is clearly $C = 1.5^2 = 2.828$ and $\gamma = -4^2 = 0.0625$.

Using this pair of parameters the final [SVM](#) is again retrained on the entire dataset \mathbf{D}_r and is now ready for testing.

In addition to the prediction output, for each sample, “LIBSVM” also produces probability estimates for the other classes. For each input sample the classifier will estimate the probability that the sample belongs in each of the possible classes. The class with the highest likelihood will be the result. The method used is an improved implementation of the work by Platt [142] presented by Lin et al. [103]. The method uses the [SVM](#) decision value (distance of sample from dividing hyperplane), predicted class and target class to establish the probability of a correct prediction given the magnitude of the decision value. The probability model is expressed as:

$$P = \frac{1}{1 + e^{A\hat{f}+B}} \quad (6.3)$$

Where P is the probability that a sample is correctly classified given, \hat{f} the decision value. The parameters A and B are used to fit the model estimated by minimising the negative log likelihood of the training data. Full details of the implementation can be found in section 8 of the LIBSVM documentation [29]. Although these probability estimates are not strictly needed for evaluating the performance of the classifier, having them makes it possible to generate the Receiver Operating Characteristic (ROC) plots in the results section.

6.2.2 Neural network configuration

Traditionally neural networks have been used to approximate arbitrary continuous functions, however it has been found they can be effectively applied to classification problems. By having a separate output node for each class to be classified, the prediction can be obtained by selecting the output with the largest excitation (assuming the network is trained using an output of 1 for positive class membership and 0 for non-membership).

One of the reasons that SVMs have been used more frequently in recent work, compared to neural networks, is their relative ease of use. As shown in the previous section, depending on the kernel function, only two parameters need to be estimated in order to produce a tuned classifier. Also SVMs have a well defined stop condition for training i.e. when the discrimination between classes has been maximised. Neural networks, on the other hand, are a flexible framework with many possible choices for the various internal functions such as gradient estimation and weight updates. Also, because neural networks can approximate any output arbitrarily well given enough hidden nodes, the problem of over-fitting must be addressed by determining an appropriate stop condition.

Since the possible fidelity of a neural network output is proportional to the number of nodes, and paths, connecting inputs to outputs, a neural network can, in theory, be trained to achieve 100% accuracy on a given training set (assuming there are no errors in the training data). However, due to the over-specialised training required to reach this accuracy on the training set, the performance on future, unseen sets, is compromised – this is known as over-fitting [96, 170]. The solution is to stop the training after the general trend of class properties has been established, instead of continuing until every sample of the training has been forced to the correct output.

Many solutions have been proposed for this problem [28, 149], some of these require several tuning parameters in addition to those already required by the neural network in general. For example the convergence stop condition (when further training no longer reduces the m.s.e. between predictions and required outputs) depends greatly on an appropriate learning rate. The rate controls how quickly weights are adjusted after each iteration to reduce output error. If it is set too high the network may converge on a local minima, too low the network is likely to be over-trained. For this reason comparisons of the solutions tend to favour non-parametric solutions such as the validation stop [87]. The benefit of the validation stop is that the learning rate can be set as low as possible (without making training time impractically long) to encourage convergence towards a global minimum, but training will stop when performance on an independent validation set drops thus, effectively, avoiding over-training.

As explained by Caruana et al. [28] the validation stop is implemented as follows. Before training begins a small random subset of the training data is held aside and not used during the error computation phase. This set is called the validation set. The neural network is now trained as normal, computing the error and adjusting the weights at the end of each iteration. However, after adjusting the weights, the classification accuracy is tested on the validation set to measure the performance on data outside the training set. If the performance drops, or fails to improve for several iterations, then training ends and

over-fitting is prevented. Using this stopping criteria the mean squared error between predictions, and targets, at each iteration is only used as an error function to adjust the weights, not as a stopping criteria.

The neural network toolbox within Matlab © has a pre-configured neural network configuration specifically designed to deal with classification problems. It can be readily configured to use the validation stop condition as described, which now leaves two more choices for the configuration; the training function and the number of hidden nodes. The training function has the role of taking the errors, measured between targets and predictions after each iteration, and adjusting the node weights to reduce them. The documentation refers to a comparison of training methods by Judd [87], based on the findings of this report the “resilient back propagation” training function [84] is recommended for classification problems.

Finally the most important factor which determines the discriminating power is the choice of hidden node count. It is generally agreed by several sources [170, 87] that having more than one hidden layer does nothing to improve performance, except in corner cases. The hidden layer sits between the input and output layers, each node in the hidden layer connects to each node in the input and output layers. Each of the 25 features listed in Figure 5.38 on page 152 will have an input node and each of the 10 classes in Table 6.1 will have an output node. The number of path weights is therefore given by $H(25 + 10)$ where H is the number of hidden nodes.

It is reasonable to assume that the more nodes used the more detailed the class separation can potentially be. However simply choosing a large number of nodes will significantly increase training time and the need for computing resources and may not perform any better than a network with less nodes. In most cases having too many hidden nodes also causes over-training, however the validation stop method used should prevent this. There is no consensus on the ideal way to choose the number of hidden nodes, it is

problem dependent. Some sources favour using sum of input and output nodes divided by two, others favour the Euclidean norm of the input and output node counts. Since there is no agreement, and it would not be optimal to just pick a number based on a recommendation used in another problem, it was decided to conduct a search for the ideal value.

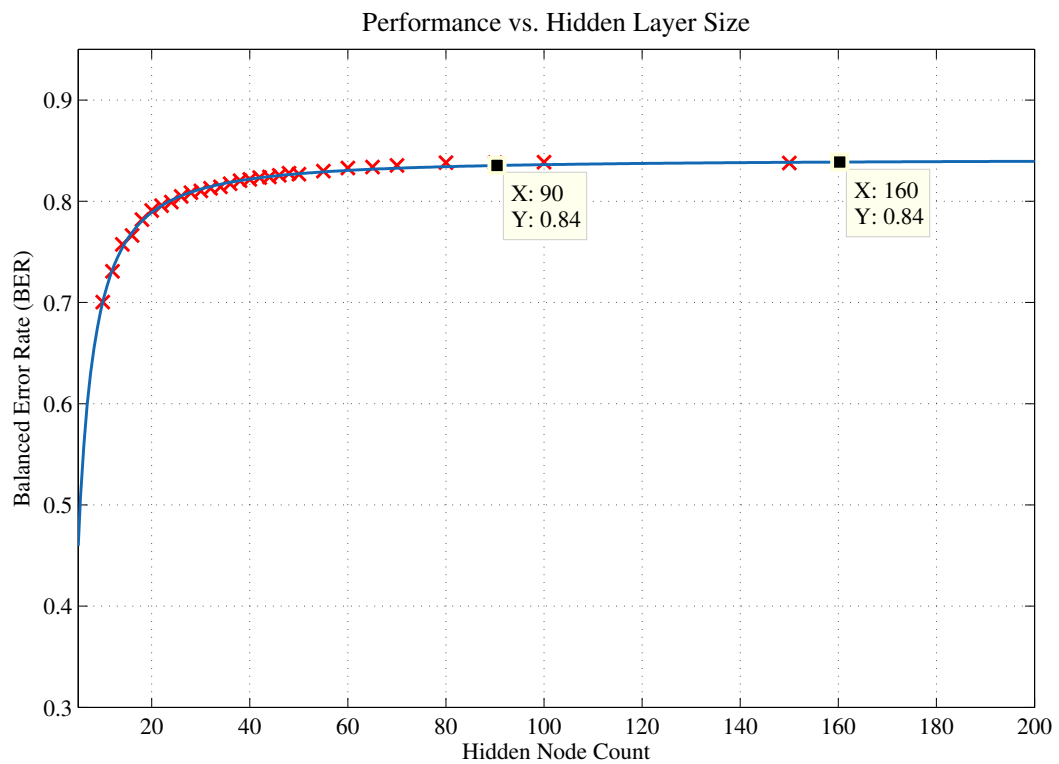


Figure 6.5: Search to find optimal number of hidden nodes, the performance appears to reach a plateau at 90 nodes so this will be the chosen value. The trend line has the form $y = ax^b + c$ and is used only as an illustration.

The goal is to find the minimum number of nodes which will maximise the network performance. Figure 6.5 shows the results of this experiment. For each hidden node count a neural network was constructed, trained on \mathbf{D}_r , and evaluated using five fold cross-validation, as was done for the SVM. Before each test a 15% validation set was separated from the training data to use as a stopping criteria. The graph clearly shows that

after 90 nodes there is no further improvement. Since the performance does not continue to increase, this result also confirms that the validation stop is preventing over-training, regardless of how many nodes are used.

In conclusion the neural network was setup as discussed using a hidden node count of 90 and a validation stop condition.

6.2.3 Comparison methodology

As discussed in [Section 6.1.2](#) classifier performance is generally evaluated using the accuracy measure i.e. correct samples divided by total samples across all classes. This is an informative measurement when the classes in the test data have equal number of members, however when the classes are imbalanced, as is the case in this research, it becomes meaningless. Some methods exist to address the issue of imbalance such as the F1 score [\[179\]](#) however it is not well suited to evaluating the overall performance of a multi-class classifier. Although it is possible to evaluate the performance of each classifier, for each class, it would be useful to have some unified scoring system to get a general performance figure.

Read and Cox [\[152\]](#) suggests a measure called the Balanced Error Rate (BER) which is essentially the average error of all classes. Here is the definition:

$$\text{B.E.R.} = \frac{1}{10} \sum_{i=1}^{10} \frac{(\sum_j M_{ij}) - M_{ii}}{\sum_j M_{ij}} \quad (6.4)$$

where 10 is the number of classes and M is the 10 by 10 confusion matrix, i.e. M_{ij} is the number of times that a sample of true class i was predicted to be in class j . For this chapter an overall performance metric based on [Equation 6.4](#) will be used which will be named the Balanced Recall Rate (BRR) given by:

$$\text{B.R.R.} = \frac{1}{10} \sum_{i=1}^{10} \frac{M_{ii}}{\sum_j M_{ij}} \quad (6.5)$$

This measure will be calculated for each classifier as a total classification score, instead of using the *accuracy*. In addition the *recall* (correctly classified samples divided by true member count, also known as the True Positive Rate - TPR and *sensitivity*) and *precision* (correctly classified samples divided by the number of samples predicted to be in a class) will be given for each class.

Because confusion matrices, and tables of figures, do not immediately convey the relative performance of a classifier ROC curves have also been produced. ROC curves provide a quick way to visually compare the performance of each classifier and the performance of each class compared to each other. It is created by plotting the recall (TPR or *sensitivity*) vs. the fraction of false positives, out of the false negatives (known as the False Positive Rate - FPR or one minus *specificity*), at various thresholds. Both the SVM and neural network give information on how likely each sample is to belong to each class in addition to the predicted class. The margin of probability between the winning class and the next best guess influence the shape of the ROC curve. The better the performance of the classifier the more tightly the curve will tuck into the top left corner of the graph (i.e. maximum true positive rate and minimum false positive rate).

6.3 TRAINING RESULTS

Please see the classifier comparison video
on the included CD or online at:

<http://tinyurl.com/TJ0-ThesisResult>

Please see [Section A.1](#) on page [226](#) for details.

The two classifiers were both trained, as discussed above, and tested on the independent testing set, D_s . Please refer to [Figure 6.12](#) on page 188, [Figure 6.13](#), [Figure 6.14](#), [Figure 7.6](#) and [Figure 7.7](#) on page 205 for examples of classified images while reading this section.

The diagonal elements of the confusion matrices (in green) show how many samples were correctly predicted to be in the respective classes, these are *true positives* (TPR or *sensitivity*). The red elements along each row show incorrectly predicted samples, the *false positives* (FPR or one minus *specificity*). The blue box in the corner shows the overall *accuracy*.

Evaluating overall performance, the comparison of [Figure 6.6](#) and [Figure 6.7](#) reveals that in this experiment the [SVM](#) classifier has a slight advantage over the neural network. The [BRR](#) indicates a 3.5% performance advantage, this result is readily visualised by comparing [Figure 6.8](#) and [Figure 6.9](#). For the [SVM](#) the [ROC](#) curve of every class clearly adheres more closely to the perfect classification profile which is a line passing through (0, 1). The lower left and lower right corners of [Figure 6.12](#) on page 188 show a side by side comparison of the two classifiers.

To confirm this in more detail [Table 6.2](#) shows that the [SVM](#) outperforms the neural network in almost every individual class, for both recall and precision. In the cases where it performs worse the difference is minimal. The only case where the neural network significantly outperforms the SVM is for “Pavement” recall, [Figure 6.6](#) reveals that the SVM has confused pavement samples with roads more often than the neural network.

Based on this outcome it would appear that for this selection of feature measurements and class distributions the [SVM](#) classifier is a better choice.

Because the same data was used to train and test the two classifiers it is logical to assume that any trends, common to both results, are independent of the classifier used. Therefore, since two different methods show a similar outcome, the root cause must lie in the

Output Class	None	Road	Sky	Pavement	Grass	Road Obj	Road Mrk	Wall	Vegetation	Pment Obj	Recall
	574 18.7%	4 0.1%	20 0.7%	7 0.2%	7 0.2%	0 0.0%	0 0.0%	12 0.4%	96 3.1%	10 0.3%	78.6% 21.4%
	18 0.6%	300 9.8%	0 0.0%	23 0.7%	4 0.1%	0 0.0%	8 0.3%	0 0.0%	2 0.1%	2 0.1%	84.0% 16.0%
	7 0.2%	0 0.0%	119 3.9%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	93.7% 6.3%
	14 0.5%	18 0.6%	0 0.0%	203 6.6%	8 0.3%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	83.2% 16.8%
	24 0.8%	2 0.1%	0 0.0%	6 0.2%	267 8.7%	0 0.0%	0 0.0%	0 0.0%	21 0.7%	2 0.1%	82.9% 17.1%
	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	38 1.2%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	95.0% 5.0%
	1 0.0%	7 0.2%	0 0.0%	4 0.1%	0 0.0%	0 0.0%	106 3.4%	0 0.0%	0 0.0%	0 0.0%	89.8% 10.2%
	14 0.5%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	208 6.8%	6 0.2%	2 0.1%	90.0% 10.0%
	98 3.2%	0 0.0%	0 0.0%	2 0.1%	7 0.2%	0 0.0%	0 0.0%	8 0.3%	657 21.4%	14 0.5%	83.6% 16.4%
	24 0.8%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	11 0.4%	84 2.7%	70.0% 30.0%
	74.2% 25.8%	90.4% 9.6%	85.0% 15.0%	82.9% 17.1%	90.8% 9.2%	97.4% 2.6%	93.0% 7.0%	91.2% 8.8%	82.6% 17.4%	73.7% 26.3%	83.1% 16.9%
Target Class											Precision
None											
Road											
Sky											
Pavement											
Grass											
Road Obj											
Road Mrk											
Wall											
Vegetation											
Pment Obj											

Figure 6.6: Confusion matrix for SVM classifier, BRR = 86.12%.

training data. Thus, in addition to comparing the performance of each classifier two experiments also allow an evaluation of the data and choice of class definitions.

Now, looking at common sources of error in both confusion matrices, it is clear, as expected, that the “None” class contributes the largest portion of error. As discussed in [Section 6.1.1](#) the “None” class has the greatest inter-class variation i.e. samples assigned to this class are not necessarily similar. The purpose of this class it to try and avoid classifying surfaces which were not present in the original training data as something they are not.

Neural Network Confusion Matrix												
Output Class	None	559 18.2%	1 0.0%	40 1.3%	2 0.1%	3 0.1%	0 0.0%	0 0.0%	36 1.2%	99 3.2%	12 0.4%	74.3% 25.7%
	Road	23 0.7%	295 9.6%	0 0.0%	16 0.5%	2 0.1%	1 0.0%	4 0.1%	0 0.0%	3 0.1%	0 0.0%	85.8% 14.2%
	Sky	9 0.3%	0 0.0%	91 3.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	90.1% 9.9%
	Pavement	22 0.7%	20 0.7%	4 0.1%	215 7.0%	12 0.4%	0 0.0%	0 0.0%	0 0.0%	6 0.2%	2 0.1%	76.5% 23.5%
	Grass	19 0.6%	4 0.1%	0 0.0%	6 0.2%	265 8.6%	0 0.0%	0 0.0%	0 0.0%	23 0.7%	2 0.1%	83.1% 16.9%
	Road Obj	1 0.0%	0 0.0%	1 0.0%	1 0.0%	0 0.0%	38 1.2%	1 0.0%	0 0.0%	0 0.0%	1 0.0%	88.4% 11.6%
	Road Mrk	2 0.1%	9 0.3%	0 0.0%	2 0.1%	0 0.0%	0 0.0%	109 3.5%	0 0.0%	0 0.0%	0 0.0%	89.3% 10.7%
	Wall	20 0.7%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	172 5.6%	9 0.3%	4 0.1%	83.5% 16.5%
	Vegetation	89 2.9%	3 0.1%	0 0.0%	3 0.1%	9 0.3%	0 0.0%	0 0.0%	16 0.5%	641 20.8%	10 0.3%	83.1% 16.9%
	Pment Obj	30 1.0%	0 0.0%	4 0.1%	0 0.0%	2 0.1%	0 0.0%	0 0.0%	4 0.1%	13 0.4%	83 2.7%	61.0% 39.0%
	Recall	72.2% 27.8%	88.9% 11.1%	65.0% 35.0%	87.8% 12.2%	90.1% 9.9%	97.4% 2.6%	95.6% 4.4%	75.4% 24.6%	80.6% 19.4%	72.8% 27.2%	80.3% 19.7%
		None	Road	Sky	Pavement	Grass	Road Obj	Road Mrk	Wall	Vegetation	Pment Obj	Precision
		Target Class										

Figure 6.7: Confusion matrix for neural network classifier, $BRR = 82.58\%$.

The same objective could be achieved by not classifying segments that do not satisfy some confidence threshold from the classifier, however this would introduce yet another tuning parameter.

It is likely that many of the segments belonging to the “None” class, that were predicted to be another class, are actually a good match for that class but were simply missed from the hand labelled data. In that case the predictions should not be considered as errors, however testing this theory would require images that have been perfectly labelled on

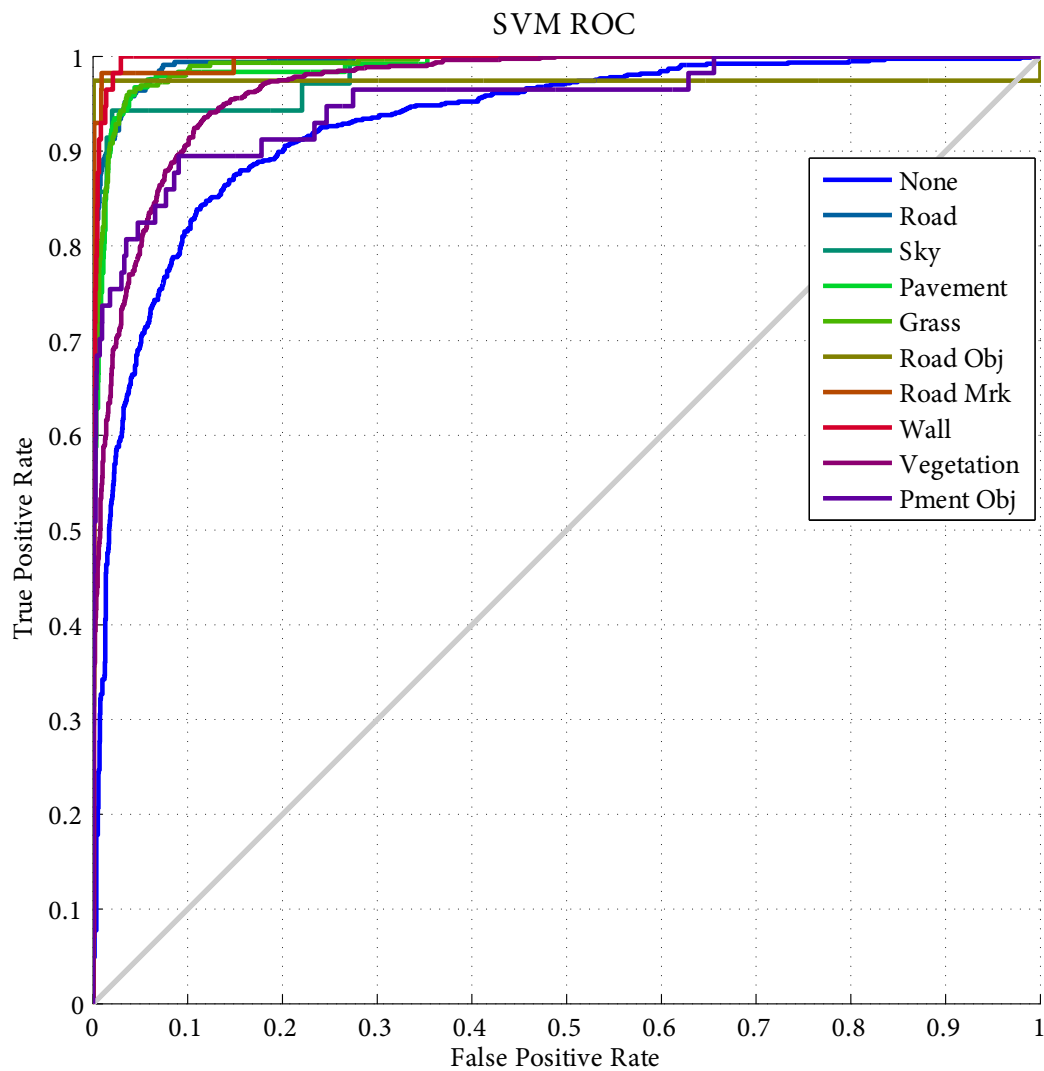


Figure 6.8: ROC curve for SVM classifier.

every pixel. [Figure 6.13](#) on page 189 shows an example of this case, without the “None” class the classifier has successfully identified “Vegetation” where the others have not. It is not always beneficial however, see [Figure 6.14](#) on page 190 for a counter example.

Another problem with having a diverse class is that ambiguous samples, which cannot be classified with great certainty are likely to end up in the “None” class instead of the next best match which could of been a correct prediction. In short the “None” class does not

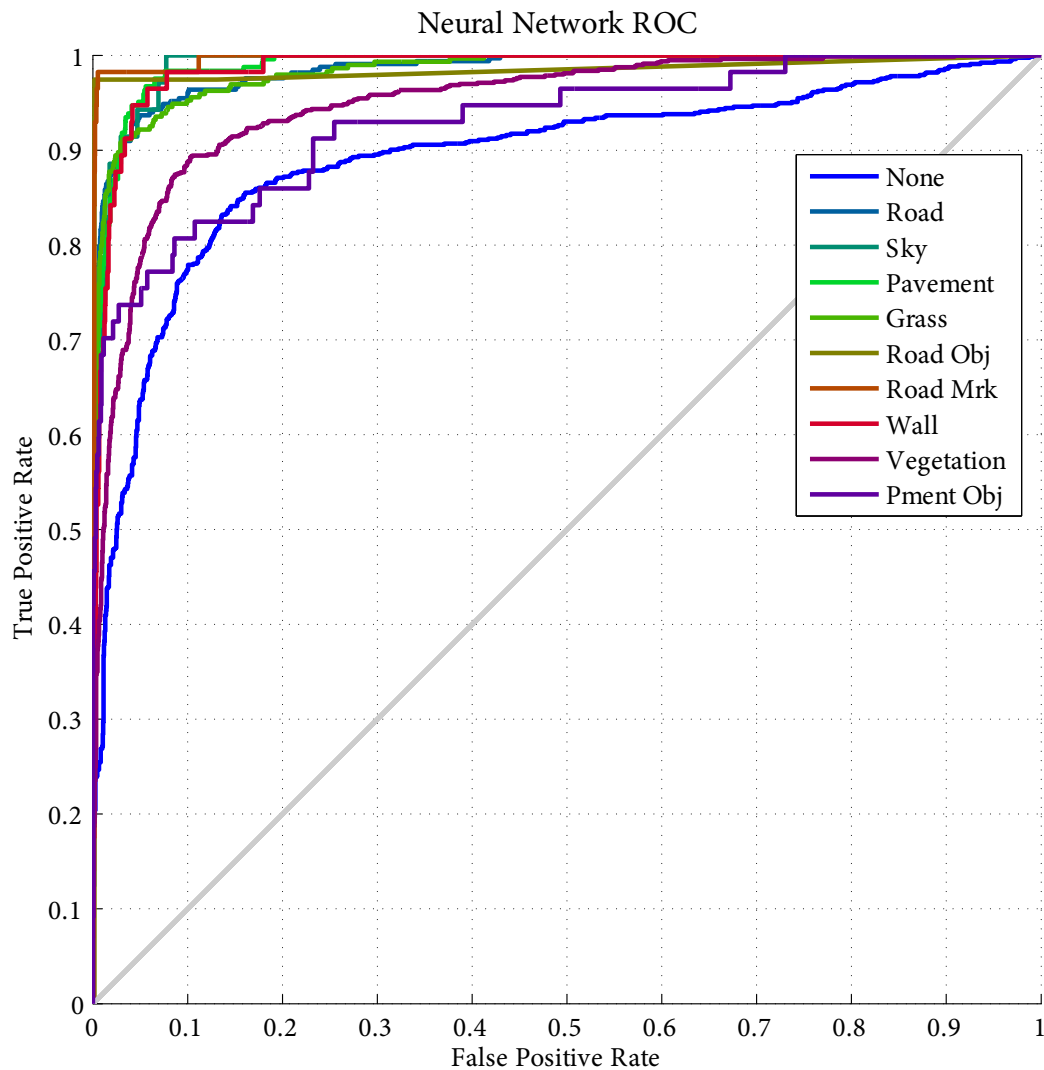


Figure 6.9: ROC curve for neural network classifier.

just decrease overall performance, because its own score is low, it also degrades other classes by attracting uncertain samples.

To better understand the impact of this on performance it was decided to take the SVM classifier (the more accurate of the two) and train it while excluding all samples in the “None” class from both the training and testing data. The result of this experiment is shown in Figure 6.10 and Figure 6.11. Table 6.2 shows performance across all classes is slightly higher. Without the low score of the “None” class bringing down the average

recall, the BRR improves by a significant 6.1%. The improvement can be immediately recognised in [Figure 6.11](#) by the tighter curve folding.

Despite the increase in overall score, the actual inter-class confusion is not significantly lower than the first experiment, so the decision on whether to use a “None” class should come down to how important it is to know when a particular surface type was not present in the training data. For example, consider a safety system which uses this classifier to identify road surfaces which are safe to drive on. If the training data did not include unsafe surfaces and there was no “None” class then any type of ground surface would be classified as the nearest known class. If it encountered a body of water it may not identify it as an unknown and therefore unsafe type of surface.

In addition to the issue of the “None” class there are other distinctive error categories common to all experiments. In particular the large confusion count between “Road” and “Pavement”, this is another expected error due to the great similarity between the classes. In fact the recall rate is unexpectedly high considering the only features separating the two are image position, region size and a small difference in height. The explanation may be that the use of the normalised horizontal position in the training data i.e. distance from the centre of the image, instead of absolute horizontal, position allows segments like pavements to be isolated more reliably. However despite this these two classes are most commonly confused with each other.

Confusion between the “Road” class and any other pose a particular problem if the system were used for applications such as a self driving vehicle. Such a system would depend on reliably identifying road regions in order to determine where it is safe to drive. If it mistakenly identifies a pavement as road surface it may result in the vehicle driving into a curb at high speed. Therefore the potential “cost” of misclassification in this case is unacceptably high and should be reduced as far as possible if this system is to be used in such a way.

SVM Without "None" Confusion Matrix											
Output Class	Road	259 11.2%	0 0.0%	11 0.5%	4 0.2%	0 0.0%	4 0.2%	0 0.0%	2 0.1%	0 0.0%	92.5% 7.5%
	Sky	0 0.0%	120 5.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	99.2% 0.8%
	Pavement	29 1.3%	0 0.0%	192 8.3%	5 0.2%	1 0.0%	2 0.1%	0 0.0%	2 0.1%	2 0.1%	82.4% 17.6%
	Grass	3 0.1%	0 0.0%	9 0.4%	286 12.4%	0 0.0%	0 0.0%	0 0.0%	48 2.1%	2 0.1%	82.2% 17.8%
	Road Obj	0 0.0%	0 0.0%	0 0.0%	0 0.0%	96 4.1%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	99.0% 1.0%
	Road Mrk	5 0.2%	0 0.0%	2 0.1%	0 0.0%	0 0.0%	110 4.8%	0 0.0%	0 0.0%	0 0.0%	94.0% 6.0%
	Wall	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	199 8.6%	7 0.3%	2 0.1%	95.7% 4.3%
	Vegetation	2 0.1%	0 0.0%	3 0.1%	5 0.2%	0 0.0%	0 0.0%	8 0.3%	745 32.2%	14 0.6%	95.9% 4.1%
	Pment Obj	0 0.0%	0 0.0%	1 0.0%	3 0.1%	0 0.0%	0 0.0%	8 0.3%	24 1.0%	98 4.2%	73.1% 26.9%
	Recall	86.9% 13.1%	100% 0.0%	88.1% 11.9%	94.4% 5.6%	99.0% 1.0%	94.8% 5.2%	92.1% 7.9%	89.9% 10.1%	83.1% 16.9%	90.9% 9.1%
		Road	Sky	Pavement	Grass	Road Obj	Road Mrk	Wall	Vegetation	Pment Obj	Precision
Target Class											

Figure 6.10: Confusion matrix for SVM classifier without “None” class, $BRR = 92.02\%$.

Similarly the classes “Vegetation” and “Grass” have an understandably high confusion rate. With the similar texture, colour and hue variations in some cases the only factor separating the two is orientation. When grass does not lie flat on the ground (e.g. a bank) then confusion is likely to occur. In this case the associated “cost” of misclassification is not significant, at least in the context of a self driving vehicle, it is unlikely that the vehicle will be depending on accurate boundaries between grass and vegetation to make safety critical navigation decisions.

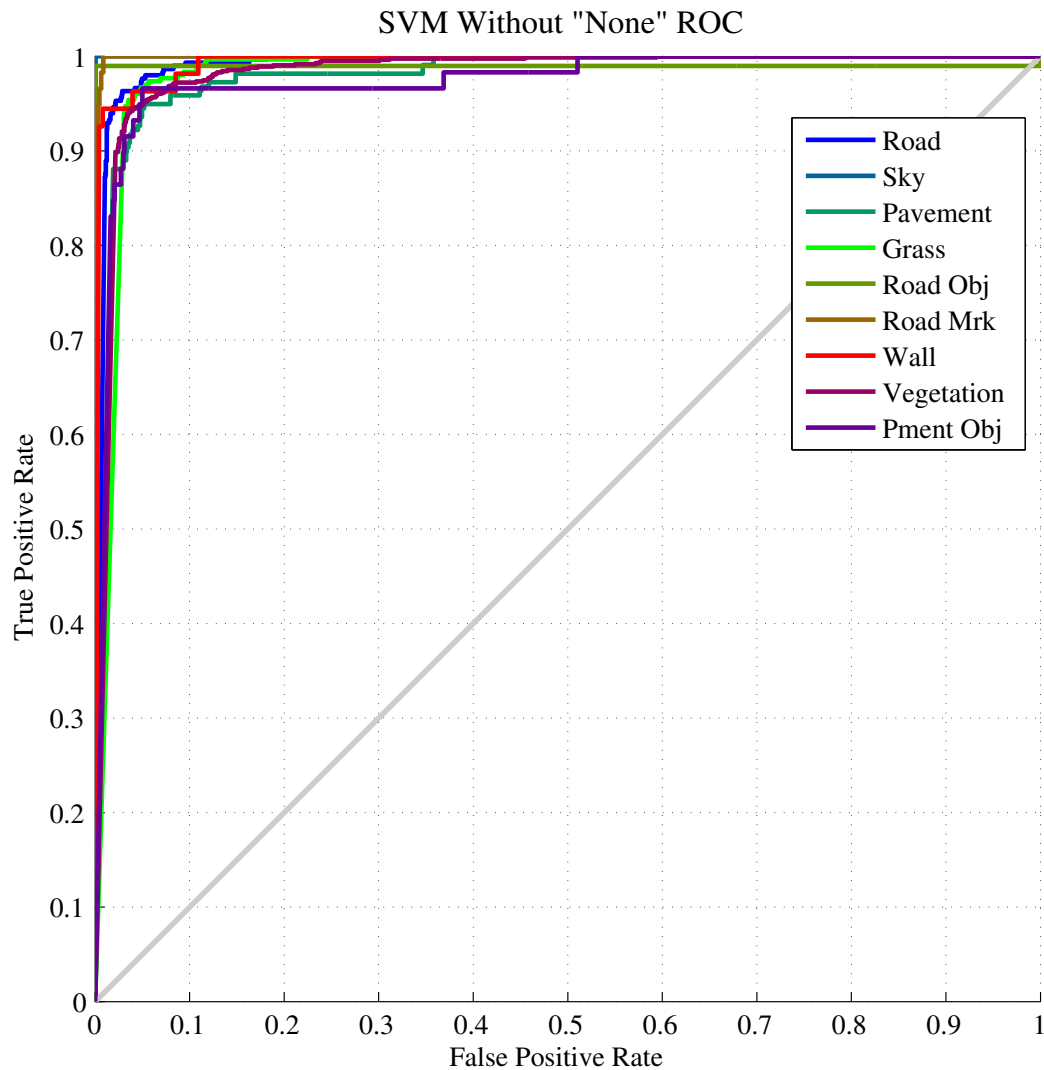


Figure 6.11: ROC curve for SVM classifier without “None” class.

The two errors, just mentioned, are almost always only confused with one other, similar looking, class. However in the case of the “Pavement Object” class the errors are distributed more randomly across the other classes. This indicates a different type of problem, the class has too few samples and too much inter-class variation. Because objects like lamp posts, bollards and signposts are not particularly similar in appearance, but are all tagged as the same class, it becomes more difficult to isolate the unique properties. It was

Table 6.2: Summary of the per-class recall (recl.) and precision (prec.) score for each experiment. The B.R.R. row gives the overall performance expressed as the balanced recall rate (mean of recall column).

CLASS	SVM		NEURAL NET		SVM W/O “NONE”	
	RECL.	PREC.	RECL.	PREC.	RECL.	PREC.
None	74.2%	78.6%	72.2%	74.3%	—	—
Road	90.4%	84.0%	88.9%	85.8%	86.9%	92.5%
Sky	85.0%	93.7%	65.0%	90.1%	100.0%	99.2%
Pavement	82.9%	83.7%	87.8%	76.5%	88.1%	82.4%
Grass	90.8%	82.9%	90.1%	83.1%	94.4%	82.2%
Road Object	97.4%	95.0%	97.4%	88.4%	99.0%	99.0%
Road Mrk.	93.0%	89.8%	95.6%	89.3%	94.8%	94.0%
Wall	91.2%	90.0%	75.4%	83.5%	92.1%	95.7%
Vegetation	82.6%	83.6%	80.6%	83.1%	89.9%	95.9%
Pment. Obj.	73.7%	70.0%	72.8%	61.0%	83.1%	73.1%
B. R. R.	86.12%		82.58%		92.02%	
Accuracy	83.1%		80.3%		90.9%	

assumed that the common location, height and size of the samples would be sufficient to distinguish them, however this experiment has not shown this to be the case.

The solution to this problem would be to define more specific classes for each type of pavement object, however this would require more effort to train and create more class size imbalance.

The only class to reach 100% recall rate (in [Figure 6.10](#)) is “Sky”. The reason is that it has by far the lowest inter-class variation. Every sample of sky is almost completely white and is the only class to have consistently missing depth information. These alone make it exceedingly easy to distinguish from other classes.

Finally the “Road Object” class shows an unexpected result. While the inter-class variation is not as high as pavement objects, not every vehicle is identical and the segments composing a vehicle are not identical in appearance. It is probable that the reason for

the high score is the large weighting it received in the training phase. Since there are only a few unique examples of road object samples, which have been heavily duplicated inside the training set, this appears to be a case of over-training. The problem reflects insufficient variation and sample count for this class in the training data. The solution would be to take more images and only label the vehicles to boost the sample count in this class.

6.4 CONCLUSION

In this chapter two classifiers, configured using best known practices, have been compared to find the most suitable tool for this task. The results show that [SVM](#) has a slight advantage in performance across all classes with no significant weaknesses compared to the neural network.

The results of the [SVM](#) show strong performance across all classes, particularly when the “None” class is excluded. Comparing this result to similar work e.g. Posner et al. [148] the experiment shows that with a significantly smaller feature space (25 features compared to 100 in Posner et al. [148] or over 6000 in Ess et al. [42]) similar, or superior, classification performance can be obtained. This is made possible by the more extensive exploitation of 3D information when analysing segment properties i.e. curved surface modelling and orientation.

Investigation into the performance impact of including a “None” class has shown that slight improvements can be made by excluding it. The drawback, however, is the loss of the ability to identify surfaces that were not explicitly trained.

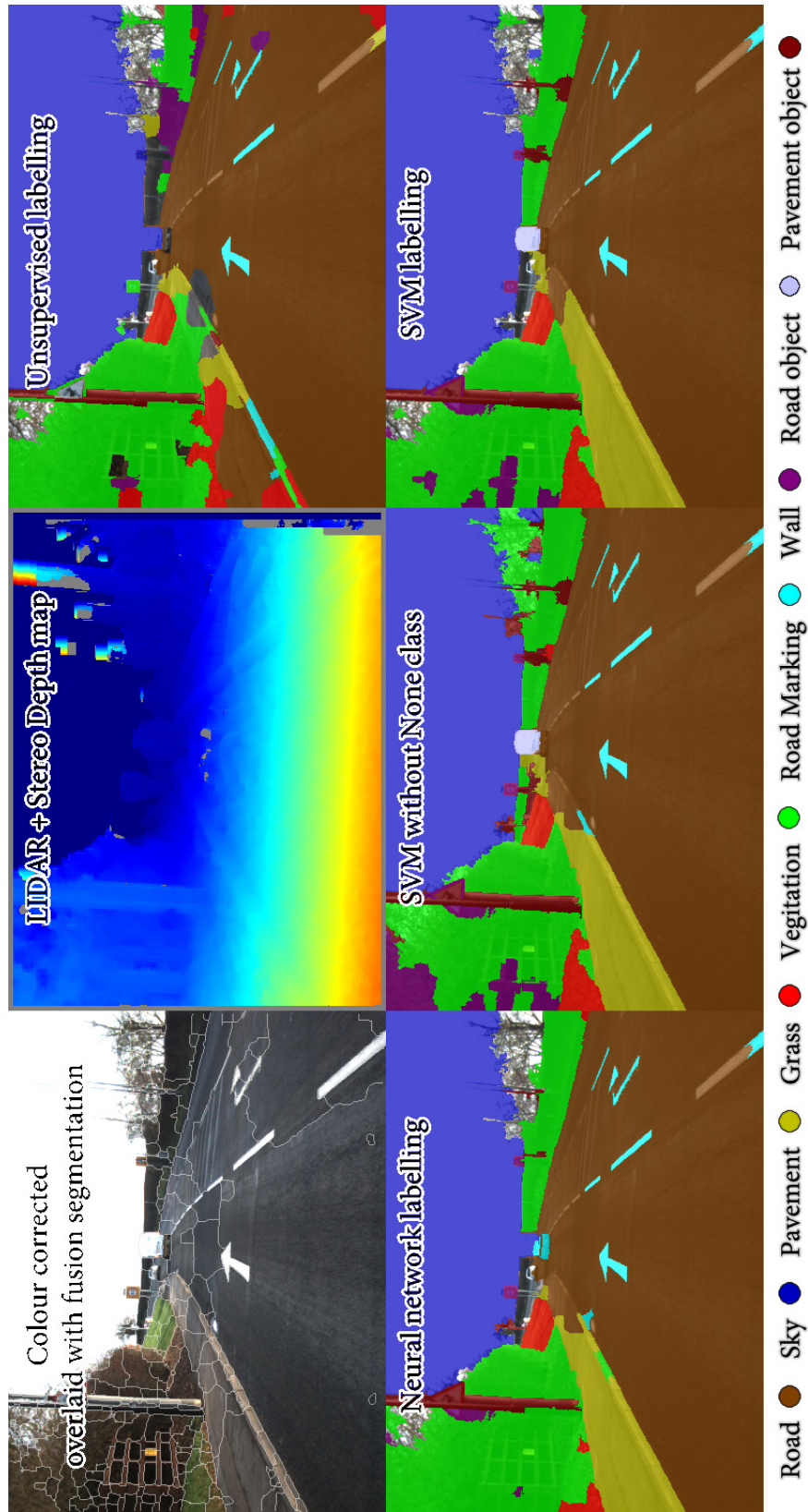


Figure 6.12: Classifier output example one, “None” class is represented by lack of shading. The three lower images relate to [Chapter 6](#), the top right one to [Chapter 7](#). In this sample it can be seen that all classifiers do well on identifying road markings. The arrow was identified even though no arrows were present in the training data. This image also shows a limitation with the segmentation. The more distant chevron lines are merged with the road because they do not satisfy minimum segment size. Also, it shows an example of an occasion when the depth map segmentation was able to separate the van from the sky. In the original image there is no separation to be found by the edge detector.

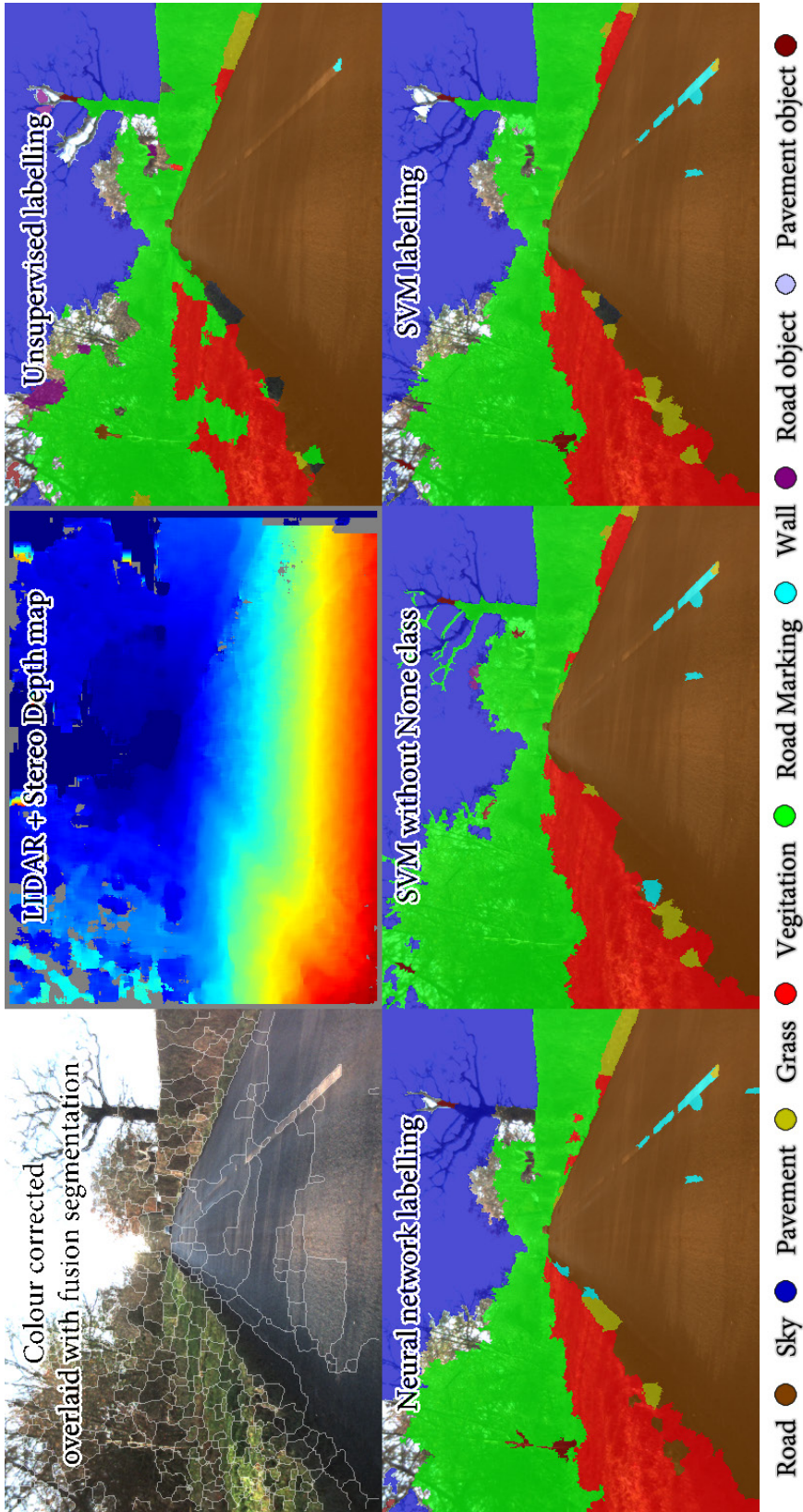


Figure 6.13: Classifier output example two, “None” class is represented by lack of shading. The three lower images relate to [Chapter 6](#), the top right one to [Chapter 7](#). This sample demonstrates an occasion when the SVM without “None” identified the areas where the trees overlap the sky as “Vegetation” which is more useful than having them identified as “None”.

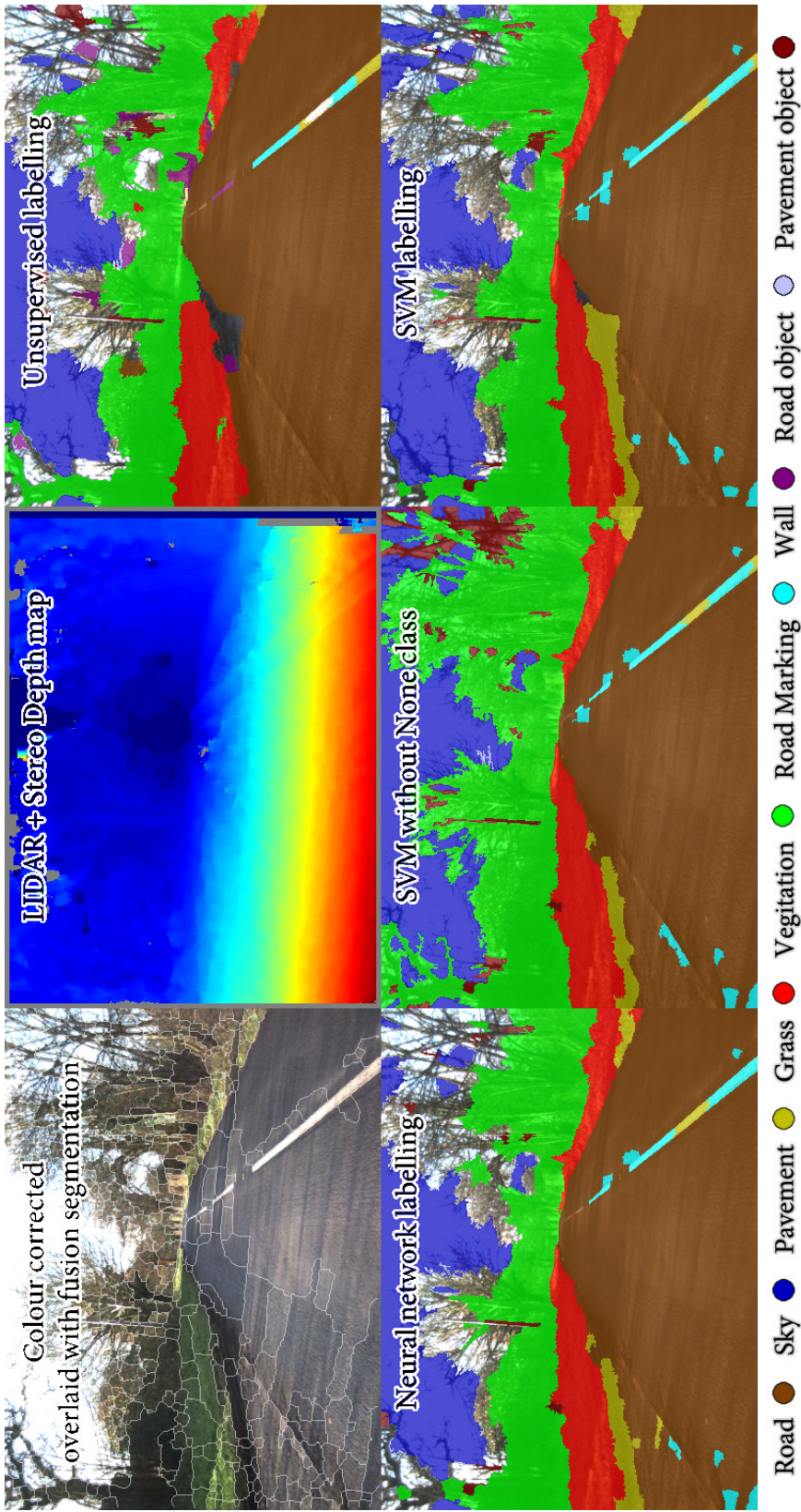


Figure 6.14: Classifier output example three, “None” class is represented by lack of shading. The three lower images relate to [Chapter 6](#), the top right one to [Chapter 7](#). The image shows a counter example to [Figure 6.13](#) showing the SVM *without* “None” classifier has identified tree foliage as “Pavement objects”. This demonstrates that removing the “None” class does not always yield better results.

7

EXPERIMENT WITH UNSUPERVISED LEARNING

CONTENTS

7.1	Experiment setup	194
7.2	Choosing the optimal number of sub-classes	196
7.3	Classification results	200
7.4	Conclusion	203

One of the foreseeable barriers, when making use of a supervised classifier such as that presented in [Chapter 6](#) is the requirement for a large quantity of training data. The time taken to label even a small group of images to the necessary accuracy is high and would need to be repeated each time the camera hardware is altered. Since it is unlikely that car manufacturers would all decide to use identical physical setups it would not be practical to share this training data. If there was a way to pre-group the segmented data into self-describing groups, then the only effort required on the part of the operator would be to give meaningful labels to these groups.

Unsupervised learning has often been used for image classification, for example work by Omran et al. [137], Fergus et al. [51], Lee and Lewicki [100], Lee and Crawford [99], Kato et al. [88]. However these methods are concerned with the classification of an entire image to identify the object it contains or a certain category to which it belongs. This method is different because it works at a lower level i.e. classifying image segments to find

the content of each region in an image. In particular it clusters segments based on content aware segmentation using a mixture of visual and spatial information. As opposed to approaches such as that by Coates et al. [33] which operates on random groups of raw pixel values for object detection.

This chapter presents the results of an experiment to test an idea for an unsupervised method. Instead of using target classes to identify differences the data will be clustered into groups which are naturally similar. In theory these groups should always contain the same type of surface due to their strongly similar features. When these groups have been formed, an operator need only observe how these groups are mapped onto a small number of sample images in order to determine how to combine these sub-groups into super-groups and how to name them.

For example, if a large number of images are segmented and automatically analysed to create an unlabelled dataset, then this dataset is automatically clustered it follows that similar looking surface types should share a cluster. These automatically determined clusters can be thought of as sub-classes which contain much more specific types of segment than the full classes, for example a sub-class might contain only a certain shade of grass on the left-hand side of the image. Looking at a single image from the set will show that areas covered in, for example, grass are always composed of the same automatically determined sub-classes. If the operator sweeps this group of sub-classes together and names the resulting super-class “Grass” then a mapping has been created that identifies grass regions in all other images. This mapping can then be applied to future images using the same clustering criteria and sub-class to super-class mappings.

Figure 7.1 illustrates the process, in this example all the sub-classes which should be mapped to “Grass” and “Road” are highlighted. The automatic sub-class creation shown in the lower left assigns a unique colour to each sub-class. The corresponding manually tagged image is included to indicate the correct mapping of super-classes. Selecting all the

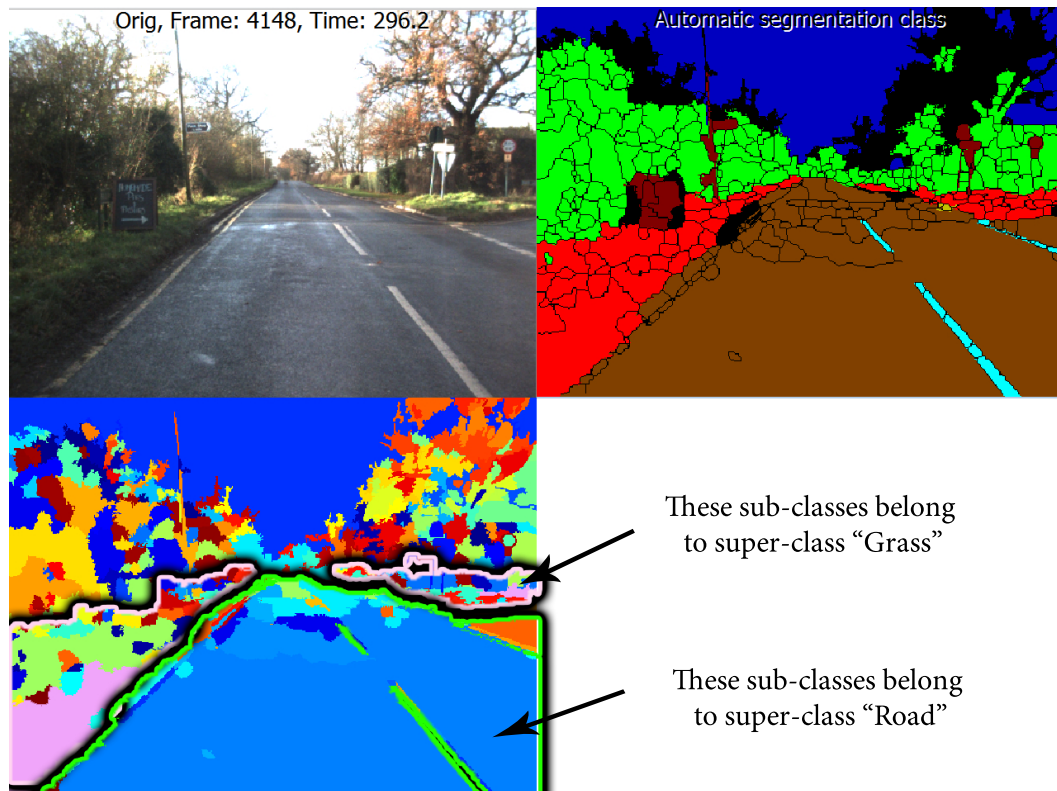


Figure 7.1: Example of sub-class to super-class mapping, top-left is the original image, top-right shows the automatic segmentation edges with manually assigned class labels, lower-left shows an example of how segments have been automatically clustered. Each sub-class has a unique colour however, due to large count, similar hues may be indistinguishable. By dragging a selection box around sub-classes that belong to the same super-class a user can create a group mapping.

sub-classes that lie in a region marked by a super-class creates a group mapping i.e. these sub-classes are all members of this super-class. Notice how, in this figure, the particular sub-classes composing the grass area are not also found in the road area. This shows that the sub-classes have separation based on the underlying image content. The particular sub-classes covering the road should, in general, be the same ones covering the road in all other images and no other type of surface.

In order to be consistent the super-classes in this experiment will be the same as the classes used in [Chapter 6](#). Since this method breaks down scenes into many sub-classes, which are independent of the choice of super-classes, other super-classes could be chosen

e.g. separate groups may be found for walls made of different materials. The number of sub-classes created will determine the fidelity of the content separation. Since the actual creation of sub-classes is automatic, and these sub-classes separate image regions based on their content, the method has been termed unsupervised. The mapping of sub-classes to super-classes is a post-processing step to make the categorisation more broad and easier to work with, the choice of these super-classes is arbitrary. Since this part does require human interaction the process could also be termed semi-supervised.

In cases where sub-classes appear in more than one super-class then it is not possible to fully separate the two super-classes. This will occur more frequently when sub-classes are not sufficiently specific to separate segments with similar appearance. This problem will be discussed further in the results section.

7.1 EXPERIMENT SETUP

For this experiment, the goal is to automatically cluster a set of segment samples into a large number of self-similar sub-classes then decide which of these sub-classes belong in each of the desired classes presented in [Chapter 6](#) which will be the chosen super-classes. The 40 hand labelled frames used in [Chapter 6](#) already have corresponding correct class assignments so the same dataset will be used to assess the performance of this method.

The 15,394 sample dataset, \mathbf{D}_a , summarised in [Table 6.1](#) on page 167 will be used to generate the clusters. Clustering is performed using the k-means algorithm [108] described in [Section 2.3.3](#) on page 36. Since the number of clusters will be determined experimentally there is no need for more complex clustering algorithms that attempt to determine the number of clusters automatically.

The data in \mathbf{D}_a are scaled to the range $[0, 1]$ as before (see [Equation 6.2](#)) however it is not up-sampled, or weighted, as this process requires the target label to be known. In

this case the data is being treated as raw data with no manual labelling, the targets will only be used to assess the accuracy of the predictions.

After clustering \mathbf{D}_a into a number of sub-classes, k , the mapping from sub-classes to super-classes would be determined by visual inspection of a small number of images tagged with the sub-classes. The images inspected to determine the mapping must contain several examples of each of the desired super-classes in order to identify the sub-classes that compose it. Several samples are needed, since a single instance of a super-class may not necessarily contain all the constituent sub-classes. Of course, if a large number of images are needed to determine the mapping of all sub-classes, then the amount of work required to assign them would defeat the purpose of this method. Therefore in this experiment a maximum of four images will be used to determine the sub-class to super-class mappings. The images are chosen such that they contain several examples of each super-class, particularly the minority classes. These images will be called the assignment images, one of them is shown in [Figure 7.1](#).

Because this experiment will be repeated many times with increasing number of sub-classes the process of inspecting and identifying which sub-classes compose a super-class will be automated using the manual labelling of these four images. The automatic assignment is done using a simple voting system. The number of times each sub-class overlaps a super-class is counted then the sub-class is assigned to the super-class with the highest count. For example, if sub-class 1 marks 50 segments manually labelled as “Grass” and 5 segments manually labelled “Vegetation”, then it is assigned to “Grass”. This automates the job of a person dragging a selection box around the correct sub-classes.

After sub-classes are assigned to super-classes a final super-class prediction, for each segment, is produced. This is then compared against the target classes obtained from manual labelling to produce a confusion matrix. The entire set \mathbf{D}_a will be used for this since there are not distinct training and testing sets.

The only parameter that must be determined for this experiment is the number of sub-classes to use, k . If the number of sub-classes is too low then there will be insufficient division in the clusters to separate similar super-classes. If there are too many then it is likely that some sub-classes will not be present in the assignment images and therefore will not be assigned to super-classes. If many sub-classes are not identified, then more than the four assignment images would be needed to fully map all sub-classes to super-classes. The more sub-classes that need assignment, the more time consuming it would be to complete the task for an operator using the system. Since it is the aim of this approach to reduce the amount of work required to produce a correct mapping, if more than the four assignment images are needed then the results will be penalised.

This is done by including unidentified sub-classes (sub-classes not present in the assignment images) in the results as class “None”. If all unassigned sub-classes are simply defaulted to the “None” class then the prediction accuracy will drop as the number of unidentified sub-classes increases. This will be explained further during the preliminary experiment to determine k .

7.2 CHOOSING THE OPTIMAL NUMBER OF SUB-CLASSES

In this section the sub-cluster count, k is determined experimentally by studying its effect on classification performance. Using a k count ranging from 10 to 500 the dataset D_a was clustered into k sub-classes. The sub-classes were then assigned to super-classes using the four assignment images (replacing the need for manual selection). The sub-class assignments were then used to map sub-classes to super-class predictions. These predictions were compared to known labelling of D_a to calculate accuracy and Balanced Recall Rate (BRR) (see Equation 6.5).

Any sub-classes that are not assigned will default to the “None” class which will degrade classification performance. This should create an upper limit on the performance of

the classification. If too many clusters are used, then the method becomes impractical. Therefore the system is constrained to making the assignments using only the four images. Both too few and too many sub-classes will produce poor classification results, therefore it is expected that the performance will peak at a particular value.

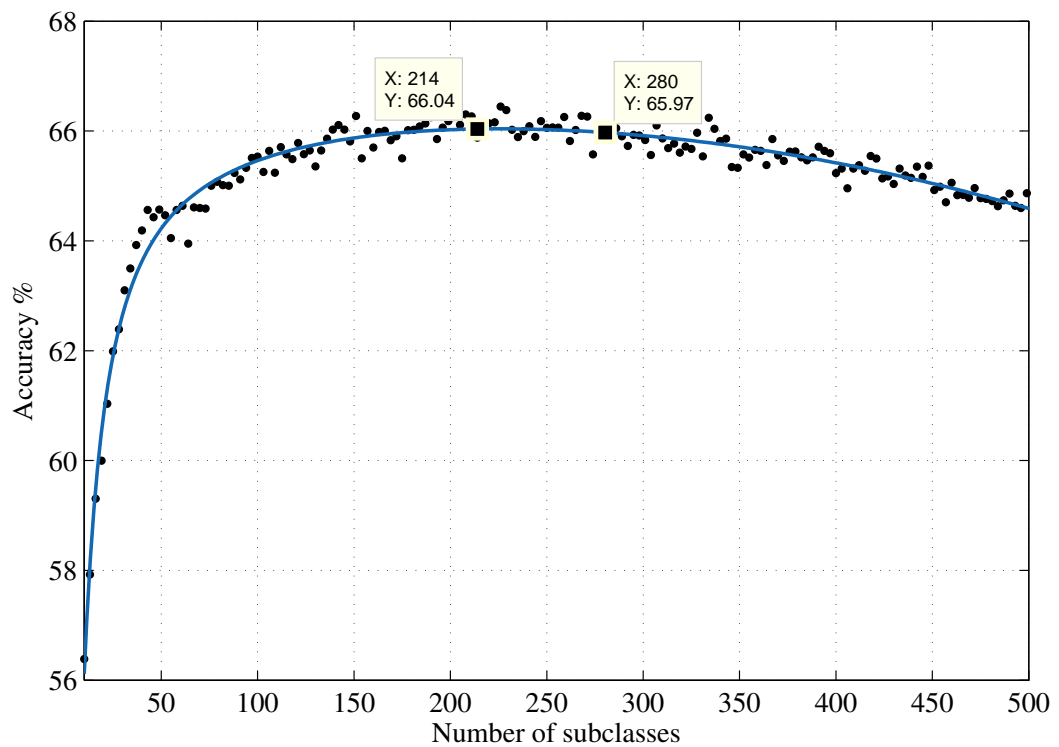


Figure 7.2: Accuracy of unsupervised method against sub-class count, the performance reaches a maximum plateau between 150 and 300 sub-classes.

Figure 7.2 and Figure 7.3 show the calculated accuracy and BRR at each sub-class count respectively. As discussed in Section 6.2.3 accuracy is a misleading measurement when classes are imbalanced, so more emphasis will be placed on Figure 7.3. As expected Figure 7.3 shows that the performance does not continue to increase unconstrained with increasing sub-classes. In fact it reaches a clearly defined maximum of 57.9% at approximately 280 sub-classes. At the far left of the graph there is the beginning of a

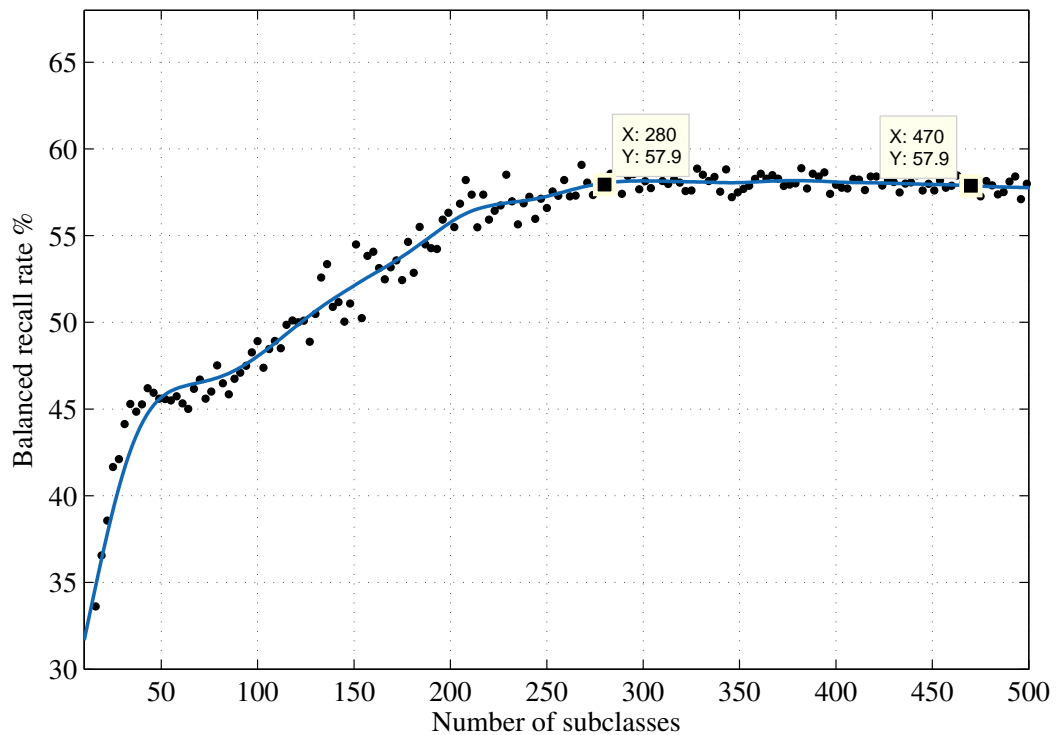


Figure 7.3: Accuracy of unsupervised method against sub-class count, the performance reaches a maximum threshold of 58% at 280 sub-classes.

downward trend indicating that performance begins to drop with further increases of k . Figure 7.2 supports this result; at 280 points the performance is within a negligible margin of the peak accuracy. This graph shows a more pronounced drop in performance as k increases to large values. These graphs together indicate that by inspecting only four images to identify sub-class mappings 57.9% is the best expected performance.

To investigate the interesting shape of Figure 7.3 the distribution of sub-classes within super-classes was also recorded. Figure 7.4 shows how many of the k sub-classes were assigned to each of the super-classes listed in the legend. Of most importance is the number of unidentified sub-classes (thick red line). The decline in performance in Figure 7.2 appears to coincide with the point in Figure 7.3 where the number of unidentified classes begins growing at an increasing rate. At the chosen count of 280 the number of unidentified sub-classes is 14 i.e. 5% of the total sub-class count. This should manifest as

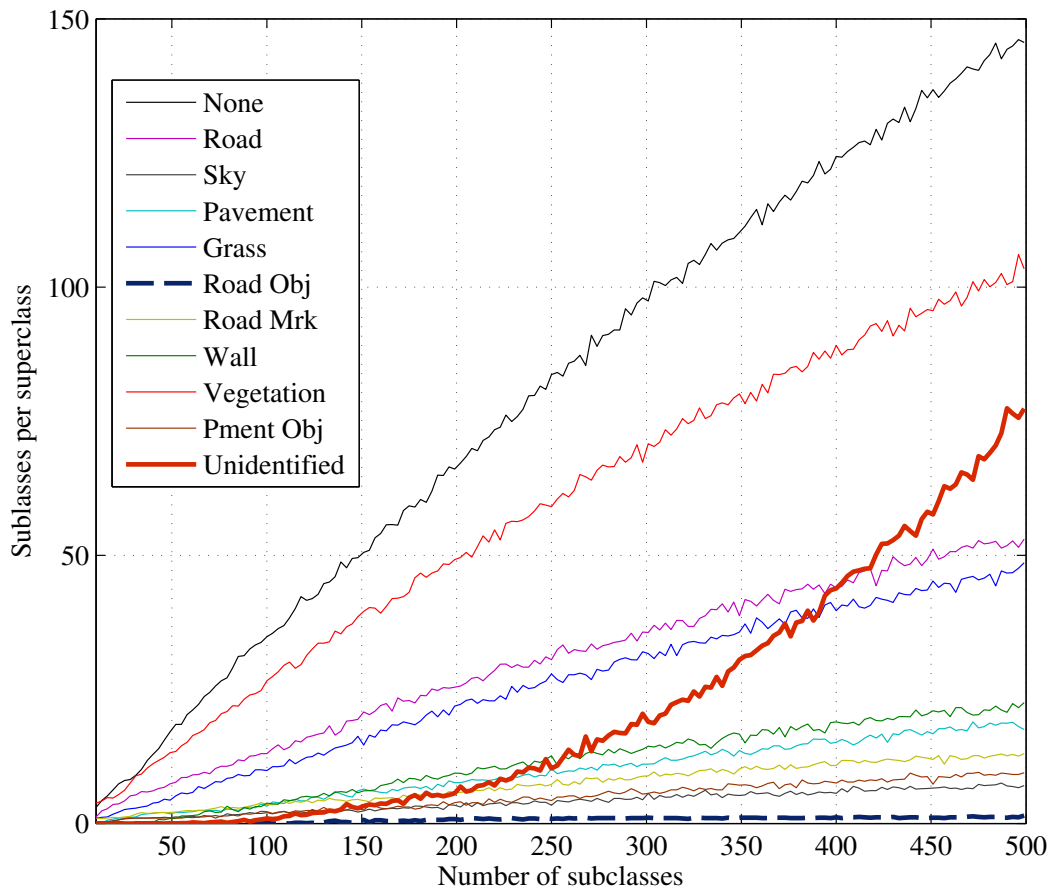


Figure 7.4: Distribution of super-class membership against sub-class count, of particular interest is the proportion of unidentified sub-classes which grow non-linearly.

5% more predicted samples in the “None” class than would be present if every sub-class was identified.

There are only a limited number of segments in the assignment images so, as k increases, the maximum potential of sub-class assignments stays constant. So as k grows the specificity of each sub-class increases, as this happens the chances of any given sub-class appearing in one of the chosen images drops. Again, it is useful to have this effect limiting the performance of the classifier, as it prevents the creation of an unwieldy number of sub-classes.

Also of interest in Figure 7.4 is the relative distribution of sub-classes. The ranking of sub-class counts does not match the ranking of class size as one might expect. The “None” class actually has over 100 fewer samples than “Vegetation” yet it has significantly more sub-classes. This is a reflection of the intra-class variation discussed in more detail in Section 6.3. The “None” class is a collection of many different types of surface so it is logical that these samples would split into a larger variety of sub-classes. This result alone provides an interesting insight into the amount of intra-class variation present in a given set.

Finally looking at “Road Object” (the dashed line) it would appear that Figure 7.3 reaches maximum performance when this line levels out. Because the class is a minority, compared to all others it appears that it does not get a share of sub-classes until the number of k is over 150. When it gets no sub-classes, then the super-class prediction will get a recall score of 0% which will degrade the BRR which is why Figure 7.3 shows such a low performance below 150 sub-classes. Since there are only a few examples of this in the assignment images, its total member count can never exceed that number, which is why it levels off. This is a limitation of the automatic assignment method.

Using a k value of 280, as determined by Figure 7.3 the experiment was run again and the predictions will now be compared to the manually labelled data.

7.3 CLASSIFICATION RESULTS

Please see the classifier comparison video
on the included CD or online at:

<http://tinyurl.com/TJ0-ThesisResult>

Please see Section A.1 on page 226 for details.

Please refer to [Figure 6.12](#) on page 188, [Figure 6.13](#), [Figure 6.14](#), [Figure 7.6](#) and [Figure 7.7](#) on page 205 for a direct comparison of classified images while reading this section.

Unsupervised Confusion Matrix												
Output Class	None	2869 18.5%	269 1.7%	144 0.9%	341 2.2%	191 1.2%	97 0.6%	50 0.3%	292 1.9%	888 5.7%	170 1.1%	54.0% 46.0%
	Road	109 0.7%	1152 7.4%	0 0.0%	224 1.4%	83 0.5%	0 0.0%	146 0.9%	28 0.2%	43 0.3%	16 0.1%	64.0% 36.0%
	Sky	50 0.3%	0 0.0%	392 2.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	24 0.2%	2 0.0%	2 0.0%	83.4% 16.6%
	Pavement	15 0.1%	63 0.4%	0 0.0%	298 1.9%	36 0.2%	0 0.0%	22 0.1%	0 0.0%	22 0.1%	2 0.0%	65.1% 34.9%
	Grass	62 0.4%	40 0.3%	0 0.0%	154 1.0%	918 5.9%	0 0.0%	2 0.0%	64 0.4%	357 2.3%	2 0.0%	57.4% 42.6%
	Road Obj	1 0.0%	4 0.0%	20 0.1%	0 0.0%	0 0.0%	308 2.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	92.5% 7.5%
	Road Mrk	45 0.3%	13 0.1%	12 0.1%	41 0.3%	1 0.0%	20 0.1%	362 2.3%	8 0.1%	3 0.0%	0 0.0%	71.7% 28.3%
	Wall	163 1.1%	8 0.1%	32 0.2%	12 0.1%	6 0.0%	0 0.0%	0 0.0%	512 3.3%	165 1.1%	52 0.3%	53.9% 46.1%
	Vegetation	563 3.6%	32 0.2%	0 0.0%	58 0.4%	236 1.5%	0 0.0%	0 0.0%	152 1.0%	2565 16.6%	80 0.5%	69.6% 30.4%
	Pment Obj	45 0.3%	1 0.0%	56 0.4%	0 0.0%	12 0.1%	0 0.0%	8 0.1%	16 0.1%	22 0.1%	210 1.4%	56.8% 43.2%
	Recall	73.2% 26.8%	72.8% 27.2%	59.8% 40.2%	26.4% 73.6%	61.9% 38.1%	72.5% 27.5%	61.4% 38.6%	46.7% 53.3%	63.1% 36.9%	39.3% 60.7%	61.9% 38.1%
		None	Road	Sky	Pavement	Grass	Road Obj	Road Mrk	Wall	Vegetation	Pment Obj	Precision
		Target Class										

Figure 7.5: Confusion matrix for unsupervised classifier, **BRR** = 57.7%.

The overall performance of the classifier expressed in terms of **BRR** is 57.7%. Compared to the Support Vector Machine (**SVM**) classifier in [Chapter 6](#) there are significantly more misclassifications. Looking at the confusion matrix in [Figure 7.5](#) it is clear that the types of error seen in the supervised classifiers are even more pronounced here.

As expected, the number of samples predicted to be in the “None” class (top row of [Figure 7.5](#)) is the greatest source of error. At least 5% of these predictions are caused by unidentified sub-classes (those sub-classes that did not appear in assignment images). Because there are many segments in the “None” class, that are almost the same as samples in named classes, the clustering did not separate the classes well. This error could be reduced by labelling a greater portion of the hand-tagged images with classes other than “None”.

Interestingly the most confused classes are the same pairs as seen in the supervised classifiers. The worst pair is “Pavement” and “Road”. As discussed, before this is due to the majority of the feature descriptors being so similar, with only three (position, region size, height) of the 25 descriptors offering any discrimination.

Because k-means weighs each of the dimensions equally, when determining the distance from a sample to a cluster centre, the 22 similar features will dominate the differences in the three features. The small number of differing features would not cause the samples to be assigned to different clusters until k is sufficiently high. It appears that some sub-classes were successfully limited to only “Pavement” or “Road” (not both) as the precision for both is still greater than 64%. To get better separation, however, more sub-classes would be needed.

This result suggests that the proposed method is not well suited to separating classes which are strongly related. Without the guidance of target classes the small differences between classes are ignored instead of exploited. The same problem can again be seen between the classes “Grass” and “Vegetation”. [Figure 7.6](#) and [Figure 7.7](#) on page 205 both show examples of these classes being confused.

[Table 7.1](#) shows the per-class performance compared to the stronger of the two supervised methods presented in [Chapter 6](#), the SVM classifier. In general it shows that the performance is stronger for classes with less intra-class variation.

Table 7.1: Summary of the per-class recall (recl.) and precision (prec.) score for the experiment compared to the SVM results from the previous chapter. The B.R.R. row gives the overall performance expressed as the balanced recall rate (mean of recall column).

CLASS	SVM		UNSUPERVISED	
	RECL.	PREC.	RECL.	PREC.
None	74.2%	78.6%	73.2%	54.2%
Road	90.4%	84.0%	72.8%	64.0%
Sky	85.0%	93.7%	59.8%	83.4%
Pavement	82.9%	83.7%	26.4%	61.5%
Grass	90.8%	82.9%	61.9%	57.4%
Road Object	97.4%	95.0%	72.5%	92.5%
Road Mrk.	93.0%	89.8%	61.4%	71.7%
Wall	91.2%	90.0%	46.7%	53.9%
Vegetation	82.6%	83.6%	61.3%	69.6%
Pment. Obj.	73.7%	70.0%	39.3%	56.8%
B.R.R.	86.12%		57.70%	
Accuracy	83.1%		61.9%	

As with the supervised classifiers the “Sky” class, with the least variation, is classified most reliably. The problem caused by insufficient samples for the class “Road Object” is apparent here too, due to insufficient samples the class has become over trained because each sample is too similar. [Figure 7.6](#) on the next page shows that the method was able to detect the “Road Object” as well as the supervised classifiers.

7.4 CONCLUSION

In this section an idea for unsupervised scenery classification was tested and evaluated. A group of 40 images were segmented, analysed and clustered into groups of segments with similar features. These groups known as sub-classes were then mapped back to the desired super-classes using only four assignment images. These images were used to observe which sub-classes are found covering each type of super-class surface. An

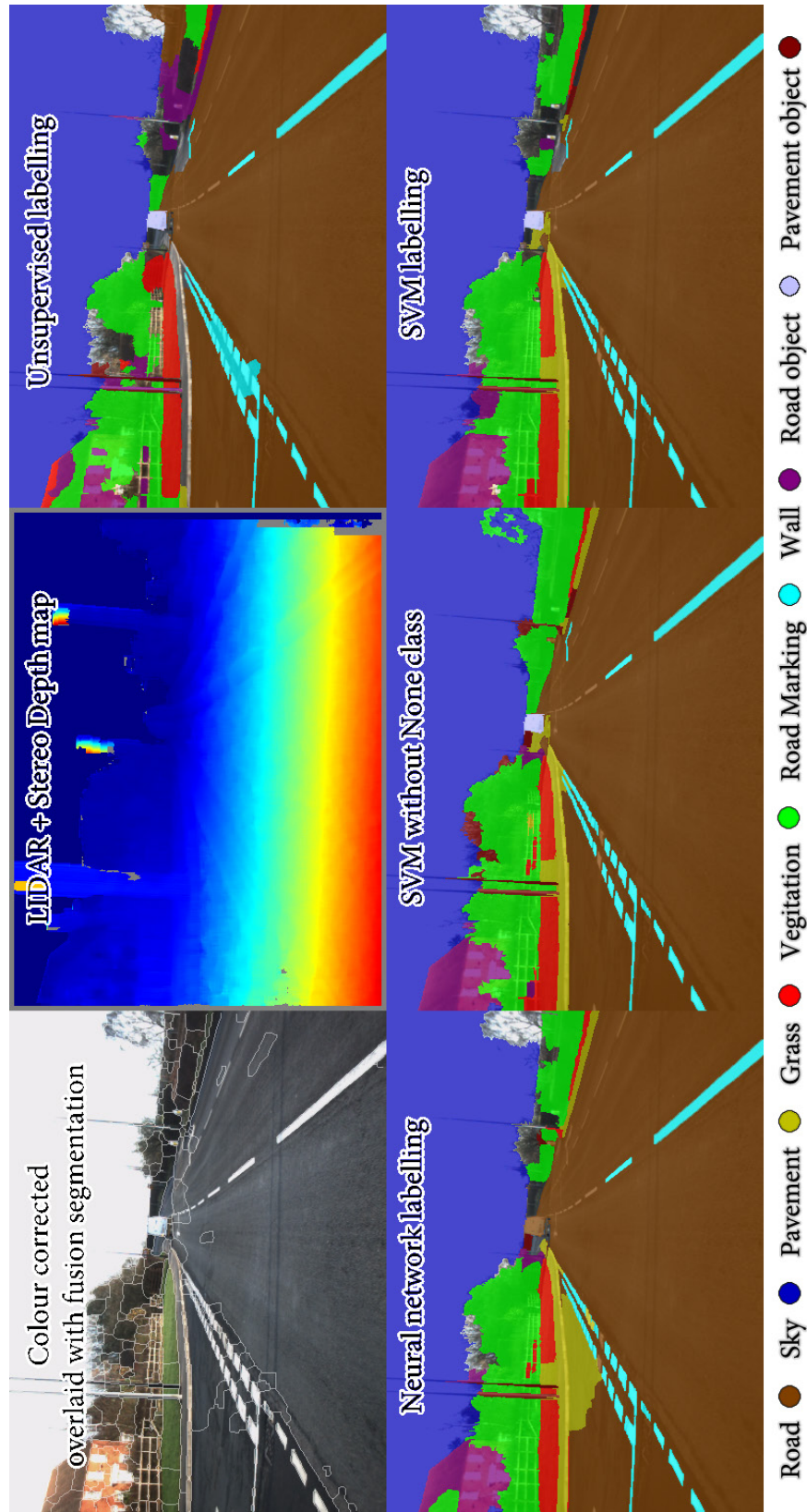


Figure 7.6: Classifier output example four, “None” class is represented by lack of shading. The three lower images relate to Chapter 6, the top right one to Chapter 7. This sample shows that despite the low score compared to the supervised classifiers the unsupervised method does a reasonable job of identifying the majority of the image. It does show confusion between “Vegetation” and “Wall” which is an error unique to the unclassified method in this image.

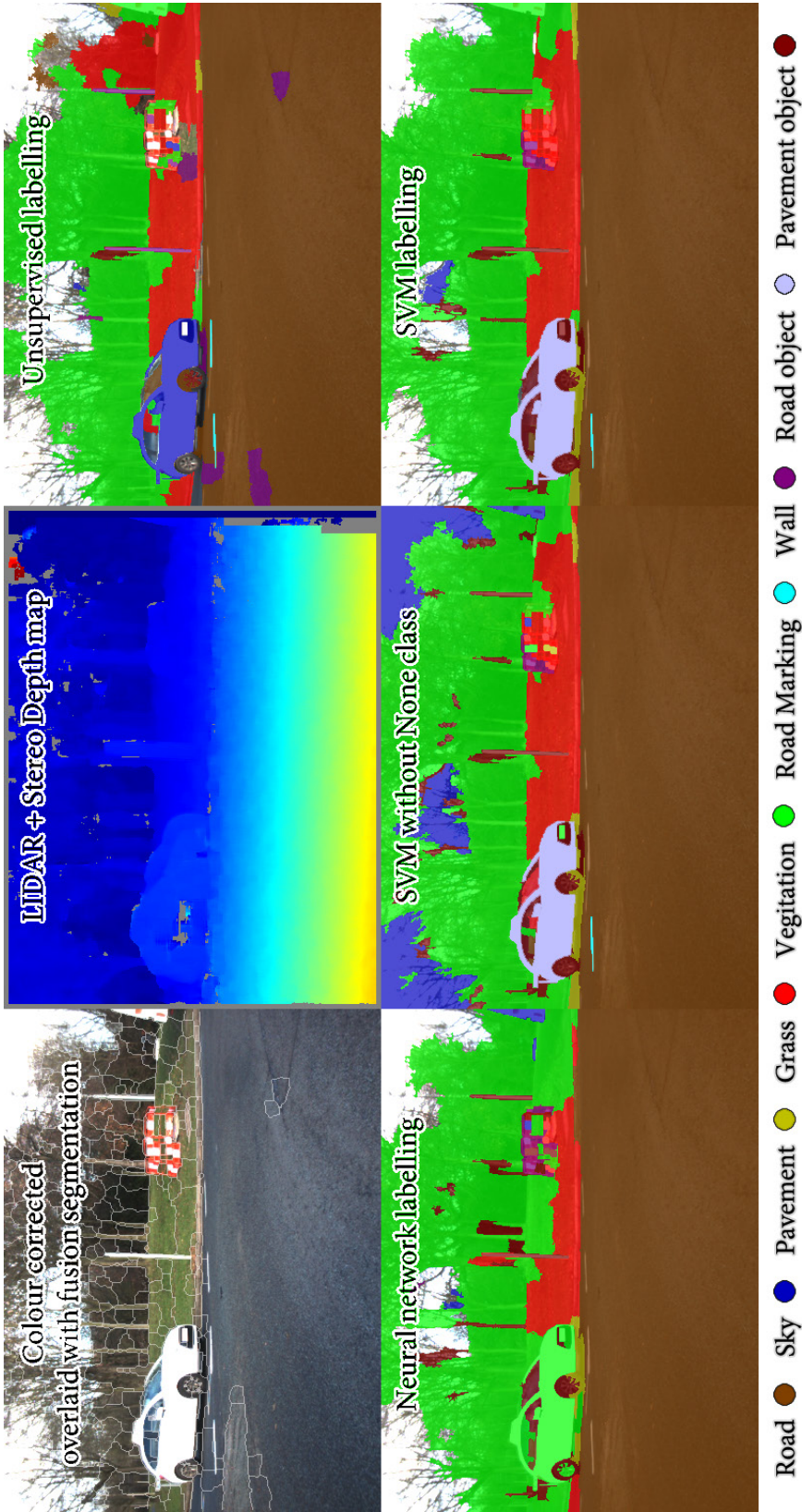


Figure 7.7: Classifier output example five, “None” class is represented by lack of shading. The three lower images relate to [Chapter 6](#), the top right one to [Chapter 7](#). This sample at a roundabout shows that a vehicle has been tagged as “Sky” by the unsupervised method due to the strong similarity in appearance. It also shows that all of the classifiers failed to mark the wheels of the car as “Road Object” – the training only has sufficient examples to identify the car body.

automatic voting procedure was used to determine which sub-classes should map to each super-class.

The optimum number of sub-classes to use was experimentally determined to be 280. This value represents a naturally occurring peak in recall performance. Using a greater number of sub-classes leads to the problem where many sub-classes cannot be identified in a small number of assignment images creating a labour intensive mapping task.

The results show that the classifier is able to achieve an average recall performance of 57.70%. The approach has been shown to suffer from a tendency to confuse classes with similar appearance. This problem is more pronounced than was observed for the supervised methods.

8

CONCLUSIONS AND FURTHER WORK

CONTENTS

8.1	Key findings and contributions	208
8.1.1	LIDAR calibration	208
8.1.2	Data collection	210
8.1.3	Segmentation and segment analysis	212
8.1.4	Comparison of classifiers based on supervised learning	217
8.1.5	Evaluation of unsupervised classifier	219
8.2	Recommendations for further work	221

In summary the selection of methods and techniques presented in this thesis accomplish all the tasks required to convert multi-sensor information into meaningful road scene classification. While it is clear that further advancements will be needed before scene labelling is sufficiently robust to be used in passenger vehicles the initial progress is promising. The supervised classifiers perform well at identifying the main types of scenery (road, vegetation, pavement and walls) this alone can provide vehicles with the ability to detect suitable driving surfaces. With this, and the inclusion of a physical control system, an Advanced Driver Assistance Systems (ADAS) could prevent the vehicle from leaving the road and colliding with the verges. Because the input data is not just imagery, but also includes physical distances an evasive manoeuvre can be planned with awareness of the real-world dimensions involved.

The potential applications for this technology span from simple driver notification systems through automatic parking and even fully autonomous driving. However before the technology can be considered for any application where human safety is concerned the prediction accuracy will need to improve even further. As mentioned in [Chapter 6](#) the current prediction results commonly mistake pavement with road regions, such a mistake in a fully autonomous vehicle could result in a high speed collision with a curb. So in its present state the system could be suitable for automating low speed manoeuvres, warning the driver about potential collisions and road surface departure given generally adequate classification performance.

The remainder of this chapter will revisit each of the previous chapters and summarise the key findings with particular focus on the contributions this body of work has made to the automated identification of driver-perspective road scenes as discussed in the introduction. In addition, several recommendations for future work and possible extensions are discussed. These recommendations generally relate to improving the performance of the discussed techniques.

8.1 KEY FINDINGS AND CONTRIBUTIONS

8.1.1 *LIDAR calibration*

[Chapter 3](#) presented a novel approach for calibrating a sensor fusion system including a camera and a laser scanner. The method employs the Nelder-Mead direct search algorithm, to achieve optimised calibration parameters, by minimising a predefined objective function.

CONTRIBUTION The novel method offers a number of improvements compared to previous methods (discussed in [Section 2.2](#)) such as simultaneous estimation of internal and external calibration parameters, no requirement for a specific test

pattern, and is thus more suitable for practical applications and for use in the field. To use this method the operator needs only 11 easily made targets that can be distributed over a wide space covering the sensor's combined Field Of View (FOV).

By recording the positions of the targets, measured by each sensor from a single pose, the 12 parameters (6 translation and angular offset, 3 intrinsic camera parameters, 3 distortion coefficients) can be estimated simultaneously. The results have sufficient accuracy to project all laser data, onto the image, such that every point is within 0.98px of its expected location (worst case) while the average error is 0.52px. This work has also been published in *Measurement science and Technology* [139].

The estimation algorithm is driven by the Nelder-Mead simplex direct search, the objective function is defined as the sum of squared errors between the image co-ordinates and the re-projected laser data. The objective function takes all the 12 parameters as arguments and calculates a value representing the error for this set of inputs. The Nelder-Mead algorithm then adjusts the parameters iteratively in an attempt to minimise the error. This optimisation method shows good robustness in all tests so far, at no point has it converged on an impossible solution even when given poor estimates, for example, if the initial distortion estimate is given as all zeros. It generally shows no further improvement after two passes and has never been seen to improve after a third.

It should be noted that in the experiment all target data points, observed by the camera were stored as integers i.e. the pixel index in the image. However projected laser scanner points are the result of a calculation and are therefore are floating point numbers which are not rounded, or truncated, before being compared to the image observations for error calculation. This means there will always be some rounding error, as the projected points are unlikely to fall exactly on an integer value even in the case of a perfect alignment.

In addition, the target size has significant impact on the calibration accuracy; ideally they should occupy a single pixel. The experiment has demonstrated that the alignment error increases with the size of sample targets. Also, it is important to consider the image distortion magnification i.e. a target that is one pixel in size, at the centre of the image, will be larger if placed near the edge of the image even if at the same distance. To correct this it may be necessary to move targets near the edges further from the camera than estimated. In all cases it is recommended to place some opposite colour contrast material behind the targets to improve visibility and accuracy of measurements.

Furthermore significant laser scatter from the targets was observed, depending on the type of material used. It was found that glossy materials produce substantial noise, due to specular reflections in the laser data, making it difficult to pinpoint the exact recorded position of the target. Using materials with a matt finish such as plain white card and some basic data filtering will greatly improve the accuracy of the measurements.

8.1.2 *Data collection*

This chapter provides a contribution which will be useful as a reference to future research teams. The discussion will save researchers time by highlighting some of the important points to consider when choosing data collection hardware.

Regarding stereo vision hardware, a new type of integrated stereo device built by *Videre Design* was evaluated. It was concluded that although the device can reduce processing burden on the computer, by performing on-board disparity matching, significant sacrifices, in terms of more important features, have been made. The device cannot provide both left and right images to the user whilst also transmitting depth information. Therefore, if a user wants to compare the depth computation, against another disparity algorithm, it is impossible. It also operates at a low resolution of 640 by 480 and has no means of synchronising its frame rate to another device.

It was found that considerably better depth information could be obtained from a pair of traditional high-resolution cameras such as the bumblebee2 from *Point Grey Research*. This package provides all that is necessary; high quality images and a software API which provides access to disparity maps. The interface is also accessible in most of the popular programming languages. The flexibility and quality outweigh the increased processing time in the context of image processing research.

At the time of writing the only practical choice of Light Detection And Ranging (LIDAR) sensor is an ibeo LUX. Built for automotive purposes, its performance characteristics are ideal for complimenting a stereo depth system. It has been widely used by many automotive research groups including the teams competing in the DARPA challenge [6]. The main limitation which precludes it from commercial use is its prohibitive cost.

A computer for recording lossless video data is recommended to have a powerful CPU and the fastest disk storage possible. If the computer has sufficient processing power to compress video using the h264 codec, this is recommended over the HFYU due to its superior lossless compression ratios. With better compression the burden on the storage device is reduced, as is the volume of data.

The most important recommendation from this chapter is to disable any automatic gain control. Generally automatic gain control will darken the image such that the sky is not overexposed, however in doing so the road and other scenery becomes darkened. This reduces the contrast and detail in the areas of interest for image analysis and hence significantly reduces the performance of the disparity algorithm. Therefore it is recommended to manually adjust the gain to maintain as much contrast as possible on the road scenery and to allow the sky to saturate if necessary.

Also, on the topic of disparity, quality it is recommended that the shutter speed is reduced as much as possible when recording high-speed driving. The “A46” dataset turned out to be mostly unusable due to the smoothing effect on the road surface caused by motion

blur. When surfaces are too smooth they lack sufficient detail to establish unique image patches in the left and right frames.

Finally the issue of being unable to synchronise the camera with the LIDAR means that the two data streams must be temporally offset manually in order to get data-frames which correspond to the same instant in time. By establishing several matching time instants, in the video and LIDAR feeds, it was possible to interpolate between them in order to get a matching LIDAR frame for any video frame. The problem with the video feed is that frames are not timestamped, so when frames are dropped there is no way of detecting the lost time and synchronisation begins to drift. With additional hardware it is possible to trigger one device from another however an easier solution would be to use a custom video recording software which can log frame timestamps to a separate file. Synchronisation would then be a simple case of matching up the timestamps.

8.1.3 *Segmentation and segment analysis*

In this chapter all stages of image preparation, segmentation and analysis are presented. Firstly the issues of colour and contrast correction are addressed. As discussed in the data collection chapter the use of manual gain control is a good first step towards keeping all input images of a consistent brightness. While this prevents image brightness fluctuating rapidly, according to localised changes in lighting, it does not balance image brightness across global variations in ambient lighting. In order to reduce the variation between images captured at midday and at dusk, for instance, the images must be contrast corrected.

This is done using the *grey world assumption* which states that given a broad sample of colour data, from an image, the average colour will be grey and the average intensity will be half the maximum. When an image is corrected using this method its current average

colour is calculated then each colour channel is independently scaled such that its new average is grey. When this is done for an entire sequence of images, with variable average brightness, the variation between them is significantly reduced. Bright images become darker and vice versa, this means that a classifier, trained on a group of images from one day, will generalise better when making predictions on images captured on another day in different lighting conditions.

Next images are segmented into smaller regions which can be individually analysed for classification. The objective of image segmentation is to divide the image such that no segment covers more than one type of scenery which is subsequently to be classified. It is preferable if this is done using the largest regions (fewest number) possible while meeting this condition in order to reduce the time required to analyse the segments. If a segment covers multiple types of surface, then the entire area will have the same classification causing some regions to be incorrectly classified.

Several existing segmentation algorithms were tested such as edge detection [130], mean-shift [32] and graph cuts [48]. It was clear that each had distinct strengths and weaknesses but none of them had the ideal combination of properties to produce the desired result. The edge detectors were effective at ensuring regions of different content are always separated, even if the line dividing them is weak. However in images with strong patterns, or texture, the result was hundreds of tiny segments, which were unsuitable for texture analysis. Adjusting the threshold on the detector in an attempt to reduce this problem invariably led to the loss of the desired separation of weakly divided regions. The correct balance cannot be obtained by adjusting the threshold. A similar problem was encountered with the mean-shift and graph cut segmenters. While they are effective at merging patterned regions into a single segment, which is desirable, they also merge regions with similar appearance, such as the road and pavement, which is not desirable.

CONTRIBUTION Since it was not possible to get the desired behaviour from a single algorithm, the proposed solution was to combine the strengths of the two most useful techniques; edge and mean-shift. The edge detection segmentation was based on a standard Canny filter with an extension to grow edges. The edges are grown until they meet with another edge thus creating closed segments. The edge detection was tuned to be particularly aggressive in order to avoid missing weak divisions between similarly coloured regions. The mean-shift step was configured to create large regions, merging together strongly patterned areas such as bushes.

With each segmentation method contributing one of the desired properties, they were fused together to produce a final segmentation strategy with the strengths of them both. This was done by replacing only the small segments in the edge detection segmentation with the large ones in the mean-shift segmentation. This leaves the large regions from the edge detection step, separating similar surfaces, and the merged regions, from the mean-shift step, containing patterned regions. The result is an accurate segmentation which separates different surfaces into large, manageable regions.

In addition to these, a third segmentation, obtained from the depth map, was used to define indisputable edges between physical discontinuities. The depth segmentation was configured to only detect large depth discontinuities, creating a sparse, but reliable, edge map of non-connected regions. This depth segmentation was merged with the result of the edge detection segmentation in order to create connections to existing edges and ensure closed regions exist. It was noted during the testing of this fusion method that the depth segmentation was often not needed, as the same edges were already found by the edge detector. Therefore it only makes a contribution in the occasional case when the edge detector misses an edge. It is recommended this step is skipped if computation time becomes an issue.

It was determined, by looking at the sparsity of depth information at the maximum range of interest (30m), that the minimum desired segment size should be 80px. Any smaller than this and there are generally insufficient points in the depth map to estimate orientation. To evaluate how effectively the fusion method fulfils the two goals of a) avoiding spanning multiple content types and b) having regions larger than 80px, a simple experiment was conducted.

A group of five images was manually segmented, then also segmented using the fusion method. Regions produced by the fusion segmentation, which spanned more than one region in the manually segmented version were counted. The number of regions smaller than 80px were also counted. The results showed that on average only 9% of regions, contained more than one content type and only 1.8% were smaller than 80px.

The next part of the chapter discusses how Hue, Saturation, Value (**HSV**) colour space was chosen over Red, Green, Blue (**RGB**) colour space. It was found during a Principal Component Analysis (**PCA**) that the red, green and blue components of segments were almost 100% correlated and so provided too much duplicate information. Also their values were dependent on both colour and brightness which is not necessary, since brightness is a separately measured segment feature. The **HSV** colour space helps to further reduce the effect of ambient lighting conditions by separating the colour information and the brightness information. Using the hue instead of the average red, blue and green components provides significantly better discrimination between segments of different colours.

Next, the texture and shape features which are used to build the segment description vector are explained.

CONTRIBUTION Of particular interest are the Gray-Level Co-occurrence Matrix (**GLCM**) measurements. Using the four descriptive statistics derived from the **GLCM** the texture of a segment can be effectively encoded without using a string of over

100 values such as the results obtained from SURF [17] or histogram encoding [192]. The use of only 25 segment features, in total, compared to the 100's used by other researchers increases the speed of classification significantly. The mapping of GLCM statistics was shown to demonstrate large variation, when applied to different types of scenery, allowing effective discrimination and classification. This method has not previously been used for the application of road imagery classification, currently it has only been applied to geographical image analysis [169, 36, 109].

In addition to the texture descriptors, feature measurements were derived from the 3D point cloud representing each of the segments in physical space. The point clouds are derived by using the segmentation to extract the appropriate part of the depth map. Firstly the orientation of the segment is determined by using a Singular Value Decomposition (SVD) to find a plane of best fit similar to the method used by Wulf et al. [188]. The section explained that the only important aspect of the orientation is the rotation relative to the ground. It is not important to know the direction the segment is facing i.e. rotation about the y-axis. For this reason the x and z rotations were combined into a single rotation measure using the euclidean norm of the two angles.

CONTRIBUTION After the orientation is found, a polynomial surface is fitted to the point cloud in order to determine the curvature of the surface. This is a useful segment feature which has not yet been exploited by similar works on image labelling. Estimating the curvature of a surface provides discrimination between surfaces that are otherwise similar, such as pavements and roads.

Pavements can be distinguished by the curved surface on their edges. The combination of high curvature and high fitting accuracy (which is also measured and recorded) indicates a smooth curved surface which is likely to be man-made. A strong curvature and low fitting accuracy indicates a rough uneven surface such as a hedge.

For this process the two options of robust, and non-robust fitting, were compared. It was found that the non-robust method is considerably faster without sacrificing significant accuracy. The choice of method again comes down to the degree of processing power available for a particular application.

Finally a [PCA](#) was performed on a group of data created using all of the described features on a set of images. The goal was to confirm that each feature was contributing unique and discriminating information which can be used to separate segment content types. This experiment showed that none of the features have significant correlation with each other, indicating that each of them contributes to the total variance.

8.1.4 *Comparison of classifiers based on supervised learning*

Because a comprehensive comparison between Support Vector Machine ([SVM](#))s and neural networks, in the field of scene labelling, could not be found in current literature, a full study was conducted in this chapter. The first part of this chapter serves as a guide to readers interested in doing similar work – outlining best practices for data selection and preparation.

First it presents the method used to manually label the training data. The manual labelling happens at the pixel level then the segment labelling is determined by the most common pixel label in each segment. This means that the training data is independent of the segmentation scheme used, allowing other methods to be compared.

Next the process of pre-processing the data is discussed. Each segment feature is independently scaled to have a unit range to remove numerical bias. It is important that any future data used with the classifier is scaled using the same transformation, such that it is scaled relative to the training data.

To deal with imbalanced classes (in terms of sample count) up-sampling is used to duplicate minority classes until they have a membership count equal to the largest class. This prevents the classifier from ignoring the minority classes in favour of maximising recall performance for the larger classes in order to get an artificially high accuracy score.

Before training the two classifiers preliminary tests were carried out to determine the optimal tuning parameters for each algorithm. For the SVM a grid search was performed to find the box constraint, C , and the kernel scaling factor, γ , which yield the best classification performance on this data. Similarly, for the neural network, a search was conducted to find the number of nodes in the hidden layer, which maximised performance, without being unnecessarily high. It is critical to configure both classifiers for best performance, on this particular dataset, if a meaningful comparison is to be conducted.

CONTRIBUTION A quantitative performance analysis on the SVM and neural network classifiers revealed that SVM is the better choice for this application. Judging by the Balanced Recall Rate (BRR) (see Equation 6.5 on page 177) the SVM outperforms the neural network classifier by 3.5%. This comparison has not yet been conducted in the context of scene labelling, only comparisons relating to the classification of geographical surveys have been presented [70, 191, 164]. This result justifies and supports the fact that SVM is used more commonly for scene labelling research whereas before it was only implied.

It is noted that the greatest source of misclassification, for both the classifiers, comes from confusion between the “None” class and others. The argument is that the “None” class should be included in order to exclude segments which the classifier was not explicitly trained to identify. This allows segments that do not fit well into a defined class to be flagged as “something else”. However, the importance of excluding uncertain predictions from the result and identifying unknown segments, depends on the application.

In order to evaluate the performance impact of including the “None” class, and to demonstrate the results obtained without it, a third experiment was run. This time the [SVM](#) is trained with the “None” samples excluded. It was also necessary to remove “None” samples from the test data since otherwise it would be impossible to evaluate the recall performance of this experiment. The results from this test showed that an additional 6.1% [BRR](#) can be obtained by removing the “None” class.

8.1.5 *Evaluation of unsupervised classifier*

This chapter presents a novel attempt at classifying road scenery using an unsupervised method. Unsupervised classification, if successful, would be the ideal method of road scene analysis. Removing the need for time-consuming manual image labelling, to train the classifiers, would bring this technique one step closer to mass-adoption.

The idea is to use k-means clustering to group segment data into naturally occurring clusters. It is assumed that each of these clusters will contain segments that are of the same type of surface, due to their strong similarity. These natural clusters are called sub-classes.

If the data is automatically categorised into sub-classes, then all that is needed is a way of combining these sub-classes into more recognisable super-classes. This can be done by observing the distribution of sub-classes on only a small number of images. Assigning sub-classes to super-classes in these images should create a mapping which is applicable to all images in the set.

It was found, through experimentation, that the ideal number of sub-classes for this dataset is 280. In general the performance should increase with sub-class count at the cost of additional work to map the sub-classes to super-classes. For this reason a maximum of four representative images were selected to be used in the test for assigning sub-classes. Sub-classes that cannot be identified in these four images are forced into the “None”

super-class thereby degrading classification performance. This score penalty, on having too many sub-classes, means the performance reaches a global maximum at a particular count.

In general it is clear that the unsupervised method presented here cannot match the performance of the supervised models tested in the previous chapter. The supervised methods outperform it in per-class accuracy by approximately 33% (the unsupervised method has a **BRR** of 57.70% compared to the **SVM BRR** of 86.12%). The experiment has shown that for this type of dataset (with similar class definitions) simply clustering the samples does not provide sufficient distinction between the samples that need to be separated.

Since there is little research on unsupervised image tagging available, other than in the field of geographical mapping [40, 54], this experiment should provide a useful comparison for researchers looking to improve the state of automatic image labelling.

CONTRIBUTION Although the method, in its current form, is not as accurate as the supervised methods it does show that reasonable results can be obtained with relatively little effort in terms of human interaction. This experiment appears to be the first attempt at using classic clustering techniques for the application of sub-image classification using a mixture of visual and spacial features (as opposed to categorising the image itself). The method presented could provide a building block on which improved methods are based.

Another useful outcome of this experiment is the study of how many natural clusters are found within each of the super-classes. [Figure 7.4](#) shows that the relative cluster count of each defined super-class is heavily dependent on the amount of variety of segments within a class. A study such as this could allow datasets to be tested for excessive intra-class variation. If it is found that one class has significantly more sub-classes than expected, then splitting it into two more specific classes may

aid classification attempts. For example the “Vegetation” class shows it contains many sub-classes reflecting its variation, splitting the class into “Tree Foliage” and “Hedge” would reduce the intra-class variations of each. This would reduce the amount of confusion with similar looking surfaces overall.

8.2 RECOMMENDATIONS FOR FURTHER WORK

This thesis covers a wide range of topics required in the process of automated road scene identification. A selection of existing techniques have been sampled and in some cases improvements have been suggested. To improve the state of scene labelling even further, improvements in the following areas would be required:

FURTHER REFINEMENT OF FEATURE SET The set of learning features used in all experiments was chosen based on intuition and the assumption that each feature contributed a distinguishable facet to the description of an image segment. Although the feature set was analysed using [PCA](#) to verify that none of the features are co-linear it was not determined how useful each of the features are in terms of classification performance. An useful avenue for further research would be to methodically re-train the classifiers using several sub-sets of the features used here to find the performance loss associated with removing each feature. This would reveal the greatest contributors to robust classification and may inspire the formulation of new features based on these findings. Any features that are not making significant contributions can be removed to improve performance, particularly ones which are expensive to compute such as the solidity feature.

REMOVE DEPENDENCE ON LIDAR Camera technology is improving fast, not only can they provide better quality images in a wider range of lighting conditions, they are also getting much cheaper. This will soon lead to the situation where stereo

vision can provide high quality depth information, even on scenes illuminated by headlights. The [LIDAR](#) device used has a high cost compared to its utility in this experiment. If the prices come down, then it would make a good supporting sensor on the passenger cars of the future.

IMPROVED AGC A simple improvement, which would make a significant improvement to recorded video quality would be an Automatic Gain Control ([AGC](#)) algorithm developed specifically for this application. All that is needed is an intelligent way of ignoring unimportant bright regions, such as the sky and strong reflections. Instead, the full range of contrast should be mapped onto the areas of interest i.e. the scenery at ground level. This is a problem already widely studied in the field, the solution is generally the use of predefined or dynamically adjusting ROI's (regions of interest) to determine the gain. This is simply not an area that was researched during this project but would make a useful addition. With this, the need for manual gain control would be removed and every frame would benefit from high quality depth information in a wide range of lighting conditions.

BETTER SEGMENTATION FIDELITY The segmentation fusion method presented here generally does well at separating content types. However, due to the enforced minimum segment size, some small details, such as distant objects, are lost. The minimum size is necessary to ensure sufficient data for surface reconstruction. In hindsight, a better way might be to perform surface reconstruction independently from the image segmentation. The surface reconstruction could then sample all depth data at the same time. Then using the segmentation, the orientation and planarity can be extracted from the pre-build set of surfaces. This would allow smaller, more detailed, segments.

MORE SPECIFIC CLASSES It was found that some classes such as "Vegetation" contained a large amount of intra-class variation. It was also noted that classes with

large intra-class variation have the greatest misclassification rate. It is recommended that classes that contain several different varieties of surface be split into more specific labels. This can be done manually, or automatically, as explained in the next point.

UNSUPERVISED CLASSIFICATION To improve the current implementation a good starting point would be to analyse the statistical distribution of each segment feature. From this it may be possible to estimate a more appropriate weighting for each of the features in the k-means distance calculation. For example if the position of a feature was weighted higher, this would help to separate the road and pavement segments.

The idea could also be used to automatically sub-class and then recombine super-classes which have excessive intra-class variation. If a hand-tagged dataset is analysed, and a particular class contains too many distinct variations of that class, then k-means could be used to break that class into a number of sub-classes. For example, the “Pavement object” class contains lamp posts and signposts which creates large interclass variation. The k-means clustering may be able to separate these into sub-classes. These sub-classes would then replace the manual labelling in the training set of a supervised classifier. When the classifier makes predictions on new data, the sub-classes will be mapped back to the expected super-class such that the sub-classing is transparent to the user. This would increase the specificity of the classes to train, and improve, the accuracy of the classifier.

TEMPORAL SMOOTHING The results shown in this thesis represent the raw output from the classifiers. A common practice in similar research projects is to post-process the predictions based on historical information. A simplistic implementation of this can be seen on the included video (see [Section A.1](#) on page 226). It only averages the classification over a few frames but makes reduces predictions

that flicker between two classes significantly. Even better results could be obtained by using a statistical framework, for example Ess et al. [42] used hidden Markov models to model the probability of a patch changing from one class to another class between time frames. Since the frames are captured a short time apart there is a high probability that a region containing a class in one frame will contain the same content in the next frame. If it an uncharacteristic change occurs then this result can be overridden with the more likely classification. It would be interesting to see how much this extension can improve the results.

ADDITIONAL POST-PROCESSING In addition to temporal smoothing other forms of post processing can be applied to the output predictions. For example, using the hand-labelled training data the frequency of each class sharing an edge with each other class can be estimated. Then if, in the prediction a patch of sky, for example, is found to be surrounded by road segments, then it is likely that this is a misclassification, since this will almost never occur in reality. If the predictions are expressed as a node graph then this type of contextual information could be implemented using belief propagation on the segments predictions [49]. An even simpler approach would be to smooth out the segment's classification based on its neighbours. If a patch of pavement is entirely surrounded by road, which is occasionally seen in the outputs of these experiments, it can be replaced with its most frequent neighbour. Whether this improves or degrades performance would need to be determined by experimentation.

An extension of this contextual concept which takes advantage of spacial information from 3D point clouds can be found in work by Anand et al. [7]. When stereo cameras are able to produce the same level of accuracy as the RGB-D cameras used in this work it will be more applicable to automotive uses.

OPTICAL FLOW Another interesting concept which has been quite popular in recent literature is optical flow. It is an algorithm which can estimate the movement of each pixel from one image to the next in a video sequence. With the addition of the stereo depth data it would be possible, using an appropriate path prediction method, to identify and track the positions of non-stationary objects in the surroundings. Once the apparent movement of static scenery is filtered out, using the known or estimated vehicle speed, any remaining objects can be flagged as in motion. This additional information would make classifying pedestrians and vehicles significantly more accurate in terms of confusion rates and boundary conformance.



APPENDIX: VIDEO DETAILS AND CODE LISTINGS

CONTENTS

A.1	Video result description	226
A.2	Objective function of calibration chapter	229
A.3	Fusion segmentation	230
A.4	Plane orientation	232

This appendix contains some potentially useful snippets of Matlab code which may be useful to others as stand alone functions without heavy dependence on external data, other custom functions or environment configuration. In addition the first section provides a description of the included result video, giving some details on how it is made and the purpose of each section.

A.1 VIDEO RESULT DESCRIPTION

The video can be found on the included CD or online at the address given in [Section 6.3](#) on page 177. A high quality version can also be downloaded from:

<http://tinyurl.com/TJ0-ThesisVideo>.

The supervised classifiers, built in [Chapter 6](#) on page 159, and the unsupervised method, presented in [Chapter 7](#) on page 191, are compared side-by-side on a selection of video

clips from two of the data sets. All classifiers are trained exactly as described in their respective chapters and for the first half of the video the output is not post-processed in any way. In the second half of the video a simplistic temporal filter is applied, read below for details. The colour coding for the classifier predictions can be seen in the centre of the video.

For the first half of the video the windows from top left to bottom right contain: The image from the left camera of the stereo rig after grey-world colour correction, the fusion segmentation edges have been laid on-top of it as white lines. The LIDAR enhanced stereo map, colour encodes distance. The output from the unsupervised method discussed in [Chapter 7](#). The output from the neural network discussed in [Section 6.2.2](#). The output from the Support Vector Machine (SVM) classifier trained without samples belonging to the “None” class as discussed in [Section 6.3](#). The output from the standard SVM classifier discussed in [Section 6.2.1](#) on page 169.

In the second half of the video the “SVM without None” classifier is removed, instead a new window is added to the bottom right showing the result of using a simple smoothing technique. The temporal smoothing uses the output from the standard SVM classifier of the current frame and the previous two frames to build an averaged classification. The SVM was used simply because it was shown to produce slightly better results than the neural network.

The *mode* of the three frames determines the smoothed output. The output does not respect segment boundaries of the current frame, it simply merges the three frames using the most common classification of each pixel. Because the time between frames is only $\frac{1}{15}$ seconds this generally does not cause a problem as the image content has not changed significantly during this time frame. The effect of this is more noticeable on the road markings (since they have the greatest apparent movement) which appear to lag behind their true position in the image. Also because the road markings and pavement objects

move to a large extent between frames the averaging effect tends to filter them out. The result is a less flickery video which is more accurate on segments that do not change but less accurate on transient items.

The primary purpose of this test was to improve the appearance of the video, it is not discussed in the main thesis since it does not influence the classification accuracy.

TIME 0:07 A sequence from the “RX residential” dataset. This clip gives a demonstration of the SVM classifier successfully identifying most of the lampposts and road signs as “Pavement Objects”. After entering the more residential area it also picks up most of the homes as “Wall” surfaces. At the start of the clip it can be seen that the strong shadows on the road occasionally cause it to confuse “Road” with “Road Markings”.

TIME 0:46 A sequence from the “Berkswell manual” dataset. This clip demonstrates the ability of the classifiers to operate in a more rural environment using the same training data. Again, road markings, lampposts, and road signs are generally detected in most of the frames. “Vegetation” and “Grass” are occasionally confused where the two regions meet and exposed branches are often interpreted as “Wall” class.

TIME 1:33 A sequence from the “Berkswell manual” dataset showing the navigation of a roundabout. None of the hand labelled images used to train the classifiers featured a roundabout. Therefore it is interesting to see how the classifiers cope with a type of road geometry which is somewhat different to the typical scene. It can be seen that in general all the classifiers do a good job of identifying the grass island of the roundabout despite the fact that this region in the image is usually covered by road.

TIME 1:50 A sequence showing the failure to detect a black vehicle. This clip shows that all classifiers consistently confuse the black vehicle ahead, with the road.

Presumably because its colour is similar to the road and because there was a lack of training examples in the hand tagged images containing black vehicles. Future training sets must contain a wider variety of vehicles to attempt to solve this issue.

TIME 2:17 The same selection of sequences showing the effect of a basic temporal smoothing filter. See the description of this above.

A.2 OBJECTIVE FUNCTION OF CALIBRATION CHAPTER

Listing A.1: Objective function of calibration optimisation, ϕ

```

1  function [sse, expu, expv, ErrorVector] = errorfun(params)
2
3  %params: 1,2,3 = Camera x,y,z, 4,5,6 = CameraRot x,y,z, 7,8,9 =
4  %adjustment x,y,z
5
6  fparams=initparam;
7
8  fparams(tmpind)=params;
9
10 cx=fparams(1);
11 cy=fparams(2);
12 cz=fparams(3);
13
14 tx=fparams(4);
15 ty=fparams(5);
16 tz=fparams(6);
17
18 ex=fparams(7);
19 ey=fparams(8);
20 ez=fparams(9);
21
22 ax=fparams(10);
23 ax2=fparams(11);
24 ax3=fparams(12);
25
26 rotx = [1 0 0; 0 cos(-tx) sin(-tx); 0 -sin(-tx) cos(-tx)];
27 roty = [cos(-ty) 0 -sin(-ty); 0 1 0; sin(-ty) 0 cos(-ty)];
28 rotz = [cos(-tz) sin(-tz) 0; -sin(-tz) cos(-tz) 0; 0 0 1];
29
```

```

30     offs=[xdata ydata zdata]'-repmat([cx cy cz]',1, datapoints);
31
32     d = rotx*roty*rotz*offs;
33
34     expu = d(1,:).*(1./d(3,:))*ez+ex;
35     expv = -d(2,:).*(1./d(3,:))*ez+ey;
36
37     P(:,1) = (expu'*2-640)/640;
38     P(:,2) = (expv'*2-480)/480;
39
40     MagP = P(:,1).^2+P(:,2).^2;
41
42     Pn(:,1)= P(:,1) .* (1+ax*MagP+ax2*MagP.^2+ax3*MagP.^3);
43     Pn(:,2)= P(:,2) .* (1+ax*MagP+ax2*MagP.^2+ax3*MagP.^3);
44
45     expu = (Pn(:,1)+1)*640/2;
46     expv = (Pn(:,2)+1)*480/2;
47
48     uerr=expu-udata;
49     verr=expv-vdata;
50
51     ErrorVector = sqrt(uerr.^2+verr.^2);
52     sse = sum(ErrorVector.^2);
53
54 end

```

A.3 FUSION SEGMENTATION

Listing A.2: Segmentation fusion

```

1  %Step 1: get image egdes
2  edgeim=edge(im,'canny',[cannyweak cannystrong],cannysigma);
3  edgeim=bwmorph(edgeim,'clean');
4
5  %Edge extension algorithm
6  edgeim=EdgeExtendm(edgeim,EdgeSearchDist,true,true);
7  labelim=bwlabel(~edgeim,4);
8  edgeim=labelim==0;
9
10 edgeim = bwmorph(edgeim,'spur',inf);
11 edgeim = bwmorph(edgeim,'clean');
12
13

```

```

14 %%Step 2: get stereo edges
15 deped=edge(depthmap,'canny',[cannyweak cannystrong],cannysigma);
16 deped=bwmorph(deped,'clean');
17
18 %Edge extension algorithm
19 deped=EdgeExtendm(deped,EdgeSearchDist,true,true);
20 labelim=bwlabel(~deped,4);
21 deped=labelim==0;
22
23 %remove noise
24 se = ones(3);
25 missingdata = depthmap==256^2;
26 noisepixel = imdilate(missingdata, se) & ~missingdata;
27 deped(noisepixel) = 0;
28
29 deped = bwmorph(deped,'spur',inf);
30 deped = bwmorph(deped,'clean');
31
32
33 %% MERGE STEREO EDGES WITH EDGE MAP
34 combedgeim=edgeim|deped;
35
36 %step 3: get edison edges
37 lab2ed = edison_wrapper(imcol,'MinimumRegionArea',SizeThresh,...
38     'RangeBandWidth',6.5,...
39     'GradientWindowRadius',5,...
40     'SpatialBandWidth',9,...
41     'MixtureParameter',.3);
42
43
44
45 %make labels
46 edgelabelim=bwlabel(~combedgeim,4);
47 meanlabelim=bwlabel(~lab2ed,4);
48
49 %Step 4: for each small region in edgeim replace label with mean-
    shiftlabel
50 meanlabelim=meanlabelim+max(edgelabelim(:))+1;
51 props=regionprops(edgelabelim,'Area','PixelIdxList');
52 areas=[props.Area];
53
54 highlightsmall = zeros(size(dedgeim));
55 edfus = edgelabelim;
56 for i = 1:length(areas)
57     if areas(i) < 80
58         edfus(props(i).PixelIdxList) = meanlabelim(props(i).PixelIdxList);
59         highlightsmall(props(i).PixelIdxList) = 2;
60     end

```

```

61 end
62
63 %remove left over edges and rebuild new edges
64 edfus=modfilt2(edfus,[2 2],1,1,0);
65 edfusedge = label2edge(edfus);
66
67 %add border to stop segments touching outside from merging
68 edfusedge([1,2,end-1,end],:) = 1;
69 edfusedge(:, [1,2,end-1,end]) = 1;
70
71 %build and clean new labelim
72 edfus=bwlabel(~edfusedge,4);
73 edfus=modfilt2(edfus,[2 2],1,1,1);

```

A.4 PLANE ORIENTATION

Listing A.3: Plane orientation

```

1  x = minicred(:,1); %ACROSS
2  y = minicred(:,2); %HEIGHT
3  z = minicred(:,3); %DISTANCE
4
5  % model  $a*x+b*y+c*z+d=0$ ,  $a^2+b^2+c^2=1$ 
6
7  A=[x,y,z,ones(length(x),1)];
8  [~,S,V]=svd(A);
9  ss=diag(S);
10 coeff=V(:,find(ss==min(ss),1)); % this may be multiple
11 coeff=coeff/norm(coeff(1:3),2);
12
13 V1=coeff(1:3)';
14 V2 = [0 1*sign(V1(2)) 0];
15
16 RV = vrrotvec(V1,V2);
17 RV(4)=-RV(4);
18
19 R = vrrotvec2mat(RV);
20
21 xrot = abs(atan(R(2,3)/R(3,3)));
22 zrot = abs(atan(R(2,1)/R(1,1)));
23
24 OrientFeature = sqrt(xrot^2+zrot^2);

```

REFERENCES

- [1] M. A. Abidi and T. Chandra. A new efficient and direct solution for pose estimation using quadrangular targets: Algorithm and evaluation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17:534–538, 1995. (Cited on page 30.)
- [2] Tinku Acharya. Integrated color interpolation and color space conversion algorithm from 8-bit bayer pattern RGB color space to 12-bit YCrCb color space, Patent US6392699, May 2002. (Cited on page 92.)
- [3] Tinku Acharya. Median computation-based integrated color interpolation and color space conversion methodology from 8-bit bayer pattern RGB color space to 12-bit YCrCb color space, Patent US6356276, March 2002. (Cited on page 92.)
- [4] Ö. G Alma. Comparison of robust regression methods in linear regression. *Int. J. Contemp. Math. Sciences*, 6(9):409–421, 2011. URL <http://m-hikari.com/ijcms-2011/9-12-2011/almaIJCMS9-12-2011.pdf>. (Cited on page 145.)
- [5] Luis Alvarez, Luis Gomez, and J. R. Sendra. Algebraic lens distortion model estimation. *Image Processing On Line*, 2010. ISSN 21051232. doi: 10.5201/ipol.2010.ags-alde. URL http://www.ipol.im/pub/algo/ags_algebraic_lens_distortion_estimation/. (Cited on page 49.)
- [6] M. Aly, J. W Burdick, V. Carson, S. Di Cairano, L. Lindzey, J. Ma, R. M Murray, R. Petras, S. Pfister, D. Rizzo, et al. Sensing, navigation and reasoning technologies for the DARPA urban challenge. 2007. (Cited on page 211.)
- [7] Abhishek Anand, Hema Swetha Koppula, Thorsten Joachims, and Ashutosh Saxena. Contextually guided semantic labeling and search for three-dimensional point clouds. *The International Journal of Robotics Research*, 32(1):19–34, 2013. URL <http://ijr.sagepub.com/content/32/1/19.short>. (Cited on page 224.)
- [8] D. Anguelov, E. Parker D. Koller, and S. Thrun. Detecting and modeling doors with mobile robots. *In Proc. of the IEEE Int. Conference on Robotics and Automation (ICRA)*, 1, 2004. (Cited on page 41.)
- [9] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Y. Ng. Discriminative learning of markov random fields for segmentation of 3D scan data. *In CVPR (2)*, 1:169–176, 2005. (Cited on page 42.)

- [10] Helder Araujo, Rodrigo L Carceroni, and Christopher M Brown. A fully projective formulation to improve the accuracy of lowe's pose-estimation algorithm. *COMPUTER VISION AND IMAGE UNDERSTANDING*, 70:227–238, 1998. URL <http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.55.5885>. (Cited on page 30.)
- [11] N. Ayache and B. Faverjon. Efficient registration of stereo images by matching graph descriptions of edge segments. *Int. J. Comput. Vision*, 1:107–131, 1987. (Cited on pages 22 and 38.)
- [12] N. Ayache and F. Lustman. Fast and reliable trinocular stereovision. *Proc. 1st Int. Conf. Comput. Vision*, 1:422–427, 1987. (Cited on pages 22 and 38.)
- [13] H. Badino, D. Huber, and T. Kanade. Integrating LIDAR into stereo for fast and improved disparity computation. In *2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, pages 405–412. IEEE, May 2011. ISBN 978-1-61284-429-9. doi: 10.1109/3DIMPVT.2011.58. (Cited on pages 74, 75, and 77.)
- [14] H. H. Baker and T. O. Binford. Depth from edge and intensity based stereo. *Proc. 7th Int. Joint Conf. Artificial Intell.*, 1:631–636, 1981. (Cited on page 39.)
- [15] Dana Harry Ballard and Christopher M. Brown. *Computer vision*. Prentice-Hall, 1982. ISBN 9780131653160. (Cited on pages 28 and 39.)
- [16] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.*, 6(1):20–29, June 2004. ISSN 1931-0145. doi: 10.1145/1007730.1007735. URL <http://doi.acm.org/10.1145/1007730.1007735>. (Cited on page 165.)
- [17] H. Bay, T. Tuytelaars, and L. V. Gool. SURF: speeded up robust features. In *Proc. of the 9th European Conference on Computer Vision (ECCV)*, 1, 2006. (Cited on page 216.)
- [18] Patrice Bellot and Marc El-Bèze. A clustering method for information retrieval. *Technical Report IR-0199, Laboratoire d'Informatique d'Avignon, France*, 1999. URL <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.40.7762&rep=rep1&type=pdf>. (Cited on page 37.)
- [19] P. Bourke. YCC colour space and image compression. URL <http://astronomy.swin.edu.au/~pbourke/colour/ycc>, 2000. URL http://paulbourke.net/texture_colour/ycc/. (Cited on page 122.)
- [20] Paul S. Bradley and Usama M. Fayyad. Refining initial points for k-means clustering. In *ICML*, volume 98, page 91–99, 1998. URL <ftp://ftp.research.microsoft.com/pub/tr/.../TR-98-36.pdf>. (Cited on page 37.)

- [21] Robert Bringhurst. *The Elements of Typographic Style*. Version 3.2. Hartley & Marks Publishers, Point Roberts, WA, USA, 2008. (Cited on page 252.)
- [22] A. Broggi, P. Cerri, P. Medici, P. P. Porta, and G. Ghisio. Real-time road signs detection. *In IVS*, 1, 2007. (Cited on page 4.)
- [23] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. *In ECCV*, 1, 2008. (Cited on page 40.)
- [24] P. Buschka and A. Saffiotti. A virtual sensor for room detection. *In Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 1, 2002. (Cited on page 41.)
- [25] J. F. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-8:679–698, 1985. URL <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4767851&navigation=1>. (Cited on pages 22, 39, and 107.)
- [26] L. Cao and L. Fei-Fei. Spatially coherent latent topic model for concurrent object segmentation and classification. *In ICCV*, 1, 2007. (Cited on page 45.)
- [27] Ingrid Carlbom and Joseph Paciorek. Planar geometric projections and viewing transformations. *ACM Comput. Surv.*, 10(4):465–502, December 1978. ISSN 0360-0300. doi: 10.1145/356744.356750. URL <http://doi.acm.org/10.1145/356744.356750>. (Cited on page 56.)
- [28] Rich Caruana, Steve Lawrence, and Lee Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. *Advances in neural information processing systems*, page 402–408, 2001. URL <http://books.google.co.uk/books?hl=en&lr=&id=Mgs2FwtgNxc&oi=fnd&pg=PA402&dq=+Neural+Network+early+stopping&ots=EJlM3kgrew&sig=4KaLSfJCTntp8Sdh1i-dT7VjuuY>. (Cited on page 173.)
- [29] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27, 2011. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>. (Cited on pages 35, 170, and 172.)
- [30] D. W. Chen, S. K. Sengupta, and R. M. Welch. Cloud field classification based upon high spatial resolution textural features: 2. simplified vector approaches. *Journal of Geophysical Research: Atmospheres* (1984–2012), 94(D12):14749–14765, 1989. URL <http://onlinelibrary.wiley.com/doi/10.1029/JD094iD12p14749/full>. (Cited on page 135.)
- [31] Yi-Wei Chen and Chih-Jen Lin. Combining SVMs with various feature selection strategies. *In Feature Extraction*, page 315–324. Springer, 2006. URL http://link.springer.com/chapter/10.1007/978-3-540-35488-8_13. (Cited on page 169.)

- [32] C.M. Christoudias, B. Georgescu, and P. Meer. Synergism in low level vision. In *16th International Conference on Pattern Recognition, 2002. Proceedings*, volume 4, pages 150–155 vol.4, 2002. doi: 10.1109/ICPR.2002.1047421. (Cited on pages 14, 40, 105, 114, and 213.)
- [33] Adam Coates, Andrew Y. Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, page 215–223, 2011. URL http://machinelearning.wustl.edu/mlpapers/paper_files/AISTATS2011-CoatesNL11.pdf. (Cited on page 192.)
- [34] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5): 603–619, May 2002. ISSN 0162-8828. doi: 10.1109/34.1000236. (Cited on page 105.)
- [35] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995. URL <http://link.springer.com/article/10.1007/BF00994018>. (Cited on pages 35 and 36.)
- [36] M. De Martino, Federico Causa, and Sebastiano B. Serpico. Classification of optical high resolution images in urban environment using spectral and textural information. In *Geoscience and Remote Sensing Symposium, 2003. IGARSS'03. Proceedings. 2003 IEEE International*, volume 1, page 467–469, 2003. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1293811. (Cited on pages 135 and 216.)
- [37] R. Deriche. Using canny’s criteria to derive a recursively implemented optimal edge detector. *Int. J. Computer Vision*, 1, 1987. (Cited on pages 39 and 105.)
- [38] B. Douillard, D. Fox, and F. Ramos. A spatio-temporal probabilistic model for multi-sensor object recognition. In *Proc. of IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)*, 1, 2007. (Cited on page 43.)
- [39] William Dumouchel and Fanny O’Brien. Integrating a robust option into a multiple regression computing environment. *Institute for Mathematics and Its Applications*, 36:41, 1991. URL <http://adsabs.harvard.edu/abs/1991IMA....36...41D>. (Cited on page 145.)
- [40] Jond Dykstra and Richard W. Birnie. Reconnaissance geologic iUlapping in cliagai hills, baluchistan, pakistan, by computer processing of l_andsat data^. 1979. URL <http://techplace.datapages.com/data/bulletns/1977-79/images/pg/00630009/1450/14900.pdf>. (Cited on page 220.)
- [41] A. Ess, B. Leibe, K. Schindler, and L. van Gool. A mobile vision system for robust multi-person tracking. In *CVPR*, 1, 2008. (Cited on page 46.)

- [42] A. Ess, T. Mueller, H. Grabner, and Gool L. van. Segmentation-based urban traffic scene understanding. In *BMVC*, September 2009. (Cited on pages xi, 45, 160, 187, and 224.)
- [43] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes challenge 2008 (VOC2008) results. In *VOC*, 1, 2008. URL <http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html>. (Cited on page 40.)
- [44] Rong-En Fan, Pai-Hsuen Chen, and Chih-Jen Lin. Working set selection using second order information for training support vector machines. *The Journal of Machine Learning Research*, 6:1889–1918, 2005. URL <http://dl.acm.org/citation.cfm?id=1194907>. (Cited on page 170.)
- [45] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: a library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008. URL <http://dl.acm.org/citation.cfm?id=1442794>. (Cited on page 34.)
- [46] Clément Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Scene parsing with multiscale feature learning, purity trees, and optimal covers. arXiv e-print 1202.2160, February 2012. URL <http://arxiv.org/abs/1202.2160>. (Cited on page 46.)
- [47] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. In *CVPR*, 1, 2004. (Cited on page 40.)
- [48] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 59(2):167–181, 2004. (Cited on page 213.)
- [49] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. *IJCV*, 70:41–54, 2006. (Cited on page 224.)
- [50] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, 1, 2003. (Cited on page 44.)
- [51] Robert Fergus, Pietro Perona, and Andrew Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, page II–264, 2003. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1211479. (Cited on page 191.)
- [52] M. A. Fishier and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartomated cartography. *Communications of the ACM*, (24):381–395, 1981. (Cited on page 30.)

- [53] Department for Transport. Reported casualties by severity, road user type and country, united kingdom, latest available year - RAS30034, September 2011. URL <http://www.dft.gov.uk/statistics/tables/ras30034/>. Table ID RAS30034. (Cited on page 1.)
- [54] Lawrence Fox and Kenneth E. Mayer. Using guided clustering techniques to analyze landsat data for mapping forest land cover in northern california. 1979. URL http://docs.lib.purdue.edu/lars_symp/300/. (Cited on page 220.)
- [55] Chris Fraley and Adrian E. Raftery. How many clusters? which clustering method? answers via model-based cluster analysis. *The computer journal*, 41(8): 578–588, 1998. URL <http://comjnl.oxfordjournals.org/content/41/8/578.short>. (Cited on page 38.)
- [56] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:23–37, 1995. (Cited on page 45.)
- [57] E. S. Gadelmawla. A vision system for surface roughness characterization using the gray level co-occurrence matrix. *NDT & E International*, 37(7): 577–588, 2004. URL <http://www.sciencedirect.com/science/article/pii/S0963869504000258>. (Cited on page 135.)
- [58] Wenshuo Gao, Xiaoguang Zhang, Lei Yang, and Huizhong Liu. An improved sobel edge detection. In *2010 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT)*, volume 5, pages 67–71, July 2010. doi: 10.1109/ICCSIT.2010.5563693. (Cited on page 39.)
- [59] Xiao-Shan Gao, Xiao-Rong Hou, and Jianliang Tang. Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):930–943, August 2003. ISSN 0162-8828. doi: 10.1109/TPAMI.2003.1217599. (Cited on page 30.)
- [60] Demetrios Gatzliolis and Hans-Erik Andersen. *A guide to LIDAR data acquisition and processing for the forests of the Pacific Northwest*. 2008. URL <http://treesearch.fs.fed.us/pubs/30652>. (Cited on page 10.)
- [61] D. M. Gavrila and S. Munder. Multi-cue pedestrian detection and tracking from a moving vehicle. *IJCV*, 73:41–59, 2007. (Cited on page 46.)
- [62] S. K Gehrig and U. Franke. Improving stereo sub-pixel accuracy for long range stereo. In *IEEE 11th International Conference on Computer Vision, 2007. ICCV 2007*, pages 1–7. IEEE, October 2007. ISBN 978-1-4244-1631-8. doi: 10.1109/ICCV.2007.4409212. (Cited on page 24.)
- [63] D. B. Gennery. Object detection and measurement using stereo vision. *Proc. ARPA Image Understanding Workshop*, 1:161–167, 1980. (Cited on page 21.)

- [64] Ibeo Automotive Systems GmbH. Ibeo LUX technical manual, 2013. URL <http://ibeo-as.com/index.php/en>. (Cited on pages 3, 8, 10, 75, and 82.)
- [65] Gene H. Golub and Christian Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, 1970. URL <http://www.springerlink.com/index/j2037t54246w4gm2.pdf>. (Cited on page 141.)
- [66] Stephen Gould, Richard Fulton, and Daphne Koller. Decomposing a scene into geometric and semantically consistent regions. In *Computer Vision, 2009 IEEE 12th International Conference on*, page 1–8, 2009. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5459211. (Cited on page 46.)
- [67] W. E. L. Grimson. A computer implementation of a theory of human stereo vision. *Phil. Trans. Royal Soc. London*, B292:217–253, 1981. (Cited on page 38.)
- [68] W. E. L. Grimson. Computational experiments with a feature-based stereo algorithm. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-7:17–34, 1985. URL <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4767615&navigation=1>. (Cited on page 38.)
- [69] W. E. L. Grimson and E. C. Hildreth. Comments on digital step edges from zero-crossings of second directional derivatives. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-7:121–126, 1985. URL <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4767628&navigation=1>. (Cited on page 39.)
- [70] Yan Guo, Kenneth De Jong, Fujiang Liu, Xiaopan Wang, and Chan Li. A comparison of artificial neural networks and support vector machines on land cover classification. In Zhenhua Li, Xiang Li, Yong Liu, and Zhihua Cai, editors, *Computational Intelligence and Intelligent Systems*, Communications in Computer and Information Science, pages 531–539. Springer Berlin Heidelberg, January 2012. ISBN 978-3-642-34288-2, 978-3-642-34289-9. URL http://link.springer.com/chapter/10.1007/978-3-642-34289-9_59. (Cited on pages 159 and 218.)
- [71] R. Hadsell, P. Sermanet, J. Ben, A. Erkan, J. Han, U. Muller, and Y. LeCun. Online learning for offroad robots: Spatial label propagation to learn long-range traversability. In *Proc. of Robotics: Science and Systems*, 1, 2007. (Cited on page 42.)
- [72] Allan Hanbury. Constructing cylindrical coordinate colour spaces. *Pattern Recognition Letters*, 29(4):494–500, 2008. URL <http://www.sciencedirect.com/science/article/pii/S0167865507003601>. (Cited on page 126.)
- [73] C. Hansen, N. Ayache, and F. Lustman. High-speed trinocular stereo for mobile-robot navigation. *Proc. NATO Adv. Res. Workshop Highly Redundant Sensor Systems*, 1, 1988. (Cited on page 22.)

- [74] M. Happold, M. Ollis, and N. Johnson. Enhancing supervised terrain classification with predictive unsupervised learning. *In Proc. of Robotics: Science and Systems*, 1, 2006. (Cited on page 43.)
- [75] R. M. Haralick. Digital step edges from zero crossing of second directional derivatives. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-6:58–68, 1984. URL <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4767475&navigation=1>. (Cited on page 38.)
- [76] Paul Hawken, Amory B Lovins, and Lovins. *Natural capitalism: creating the next industrial revolution*. Little, Brown and Co., Boston, 1999. ISBN 0316353167 9780316353168. (Cited on page xvii.)
- [77] D. Hoiem, A.A. Efros, and M. Hebert. Geometric context from a single image. *In ICCV*, 1, 2005. (Cited on page 40.)
- [78] Paul W. Holland and Roy E. Welsch. Robust regression using iteratively re-weighted least-squares. *Communications in Statistics-Theory and Methods*, 6 (9):813–827, 1977. URL <http://www.tandfonline.com/doi/abs/10.1080/03610927708827533>. (Cited on page 145.)
- [79] R. Horaud, B. Conio, O. Le Boulleux, and B. Lacolle. An analytic solution for the perspective 4-point problem. pages 500–507. IEEE Comput. Soc. Press. ISBN 0-8186-1918-X. doi: 10.1109/CVPR.1989.37893. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=37893>. (Cited on page 30.)
- [80] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):415–425, 2002. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=991427. (Cited on pages 168, 169, and 170.)
- [81] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. *A practical guide to support vector classification*. 2003. URL <https://www.cs.sfu.ca/people/Faculty/teaching/726/spring11/svmguide.pdf>. (Cited on pages 36 and 170.)
- [82] Yingping Huang, Shan Fu, and Chris Thompson. Stereovision-based object segmentation for automotive applications. *EURASIP Journal on Advances in Signal Processing*, 2005(14):910950, August 2005. ISSN 1687-6180. doi: 10.1155/ASP.2005.2322. URL <http://asp.eurasipjournals.com/content/2005/14/910950>. (Cited on page 106.)
- [83] Zhexue Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2 (3):283–304, 1998. URL <http://link.springer.com/article/10.1023/A:1009769707641>. (Cited on page 37.)

- [84] Christine Hubert. Pattern completion with the random neural network using the RPROP learning algorithm. In *Systems, Man and Cybernetics, 1993. 'Systems Engineering in the Service of Humans', Conference Proceedings., International Conference on*, page 613–617, 1993. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=384942. (Cited on page 174.)
- [85] Point Grey Research Inc. Bumblebee2 technical manual, 2013. URL <http://ww2.ptgrey.com/stereo-vision/bumblebee-2>. (Cited on page 80.)
- [86] George H. Joblove and Donald Greenberg. Color spaces for computer graphics. In *ACM SIGGRAPH Computer Graphics*, volume 12, page 20–25, 1978. URL <http://dl.acm.org/citation.cfm?id=807362>. (Cited on page 125.)
- [87] J. Stephen Judd. *Neural network design and the complexity of learning*. The MIT Press, 1990. URL http://books.google.co.uk/books?hl=en&lr=&id=DOGQ_EHRXlkC&oi=fnd&pg=PP19&dq=+Neural+Network+Design,+Boston,&ots=14THentxSY&sig=N8-0PsEZ-UoLBYYH3_Hh4zKv5dM. (Cited on pages 173 and 174.)
- [88] Zoltan Kato, Josiane Zerubia, and Marc Berthod. Unsupervised parallel image classification using markovian models. *Pattern Recognition*, 32(4): 591–604, 1999. URL <http://www.sciencedirect.com/science/article/pii/S0031320398001046>. (Cited on page 191.)
- [89] Y. C. Kim and J. K. Aggarwal. Positioning 3-d objects using stereo images. *IEEE J. Robotics and Automation*, RA-3:361–373, 1987. (Cited on page 38.)
- [90] Stefan Knerr, Léon Personnaz, and Gérard Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In *Neurocomputing*, page 41–50. Springer, 1990. URL http://link.springer.com/chapter/10.1007/978-3-642-76153-9_5. (Cited on page 169.)
- [91] Ulrich H.G. Kreßel. Pairwise classification and support vector machines. In *Advances in kernel methods*, page 255–268, 1999. URL <http://dl.acm.org/citation.cfm?id=299108>. (Cited on page 169.)
- [92] B. Kuipers and P. Beeson. Bootstrap learning for place recognition. In *Proc. of the Eighteenth National Conference on Artificial Intelligence (AAAI-02)*, 1, 2002. (Cited on page 41.)
- [93] B. Kuipers and Y.T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, 8:47–63, 1991. (Cited on page 41.)
- [94] Jeffrey C. Lagarias, James A. Reeds, Margaret H. Wright, and Paul E. Wright. Convergence properties of the nelder-mead simplex algorithm in low dimensions.

- SIAM JOURNAL OF OPTIMIZATION*, 9:112–147, 1996. URL <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.56.7081>. (Cited on pages 48, 52, and 53.)
- [95] Edmund Y. Lam. Combining gray world and retinex theory for automatic white balance in digital photography. In *Consumer Electronics, 2005.(ISCE 2005). Proceedings of the Ninth International Symposium on*, page 134–139, 2005. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1502356. (Cited on page 103.)
- [96] Steve Lawrence, C. Lee Giles, and Ah Chung Tsoi. Lessons in neural network training: Overfitting may be harder than expected. In *AAAI/IAAI*, page 540–545, 1997. URL <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.38.6468&rep=rep1&type=pdf>. (Cited on page 173.)
- [97] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 1, 2006. (Cited on page 44.)
- [98] Jonathan Lee, Ronald C. Weger, Sailes K. Sengupta, and Ronald M. Welch. A neural network approach to cloud classification. *Geoscience and Remote Sensing, IEEE Transactions on*, 28(5):846–855, 1990. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=58972. (Cited on page 135.)
- [99] Sanghoon Lee and Melba M. Crawford. Unsupervised multistage image classification using hierarchical clustering with a bayesian similarity measure. *Image Processing, IEEE Transactions on*, 14(3):312–320, 2005. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1395986. (Cited on page 191.)
- [100] Te-Won Lee and Michael S. Lewicki. Unsupervised image classification, segmentation, and enhancement using ICA mixture models. *Image Processing, IEEE Transactions on*, 11(3):270–279, 2002. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=988960. (Cited on page 191.)
- [101] B. Limketkai, L. Liao, and D. Fox. Relational object maps for mobile robots. In *L. P. Kaelbling and A. Saffiotti*, 1:1471–1476, 2005. (Cited on page 41.)
- [102] Hsuan-Tien Lin and Chih-Jen Lin. A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods. *submitted to Neural Computation*, page 1–32, 2003. URL <http://home.caltech.edu/~htlin/publication/doc/tanh.pdf>. (Cited on page 36.)
- [103] Hsuan-Tien Lin, Chih-Jen Lin, and Ruby C. Weng. A note on platt’s probabilistic outputs for support vector machines. *Machine learning*, 68(3):267–276, 2007. URL <http://link.springer.com/article/10.1007/s10994-007-5018-6>. (Cited on page 171.)

- [104] Ce Liu, Jenny Yuen, and Antonio Torralba. Nonparametric scene parsing: Label transfer via dense scene alignment. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, page 1972–1979, 2009. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5206536. (Cited on page 46.)
- [105] Ce Liu, Jenny Yuen, and Antonio Torralba. Nonparametric scene parsing via label transfer. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(12):2368–2382, 2011. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5936073. (Cited on page 46.)
- [106] Yi Liu and Yuan F. Zheng. One-against-all multi-class SVM classification using reliability measures. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 2, page 849–854, 2005. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1555963. (Cited on page 169.)
- [107] C. P Lu, G. D Hager, and E. Mjolsness. Fast and globally convergent pose estimation from video images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):610–622, June 2000. ISSN 0162-8828. doi: 10.1109/34.862199. (Cited on page 30.)
- [108] James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, page 14, 1967. URL http://books.google.co.uk/books?hl=en&lr=&id=IC4Ku_7dBFUC&oi=fnd&pg=PA281&dq=MacQueen,+1967&ots=nMZeGVHaq0&sig=lg5uncdrC-kjW3DW2bfhnurFtBs. (Cited on pages 37 and 194.)
- [109] Danielle J. Marceau, Philip J. Howarth, Jean-Marie M. Dubois, and Denis J. Gratton. Evaluation of the grey-level co-occurrence matrix method for land-cover classification using SPOT imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 28(4):513–519, 1990. URL <http://www.cabdirect.org/abstracts/19940601842.html>. (Cited on pages 135 and 216.)
- [110] D. Marr and E. Hildreth. Theory of edge detection. *Proc. Royal Soc. London*, B207: 187–217, 1980. (Cited on pages 22 and 38.)
- [111] D. Marr and T. Poggio. A computational theory of human stereo vision. *Proc. Royal Soc. London*, B204:301–328, 1979. (Cited on page 27.)
- [112] Jose L. Marroquin and Federico Girosi. Some extensions of the k-means algorithm for image segmentation and pattern classification. Technical report, DTIC Document, 1993. URL <http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA271691>. (Cited on page 37.)

- [113] O. Martinez-Mozos, C. Stachniss, and W. Burgard. Supervised learning of places from range data using adaboost. *In Proc. of the Int. Conference on Robotics and Automation (ICRA)*, 1:1742–1747, 2005. (Cited on page 41.)
- [114] Timothy Masters. *Practical neural network recipes in C++*. Morgan Kaufmann, 1993. URL http://books.google.co.uk/books?hl=en&lr=&id=7Ez_Pq0sp2EC&oi=fnd&pg=PR17&dq=neural+network+tune&ots=e-5BmARmHP&sig=kKuBs3jQXnrJElT9SP9zKllpYt4. (Cited on page 168.)
- [115] M. Maurette. Mars rover autonomous navigation. *Autonomous Robots*, 14(2-3):199–208, 2003. URL <http://link.springer.com/article/10.1023/A:1022283719900>. (Cited on page 4.)
- [116] J. E. W. Mayhew and J. P. Frisby. Psychophysical and computational studies towards a theory of human stereopsis. *Artificial Intell.*, 17:349–385, 1981. (Cited on page 38.)
- [117] J. C. McCall and M. M. Trivedi. Video-based lane estimation and tracking for driver assistance: survey, system and evaluation. *ITS*, 1, 2006. (Cited on page 4.)
- [118] G. Medioni and R. Nevatia. Segment-based stereo matching. *Comput. Vision, Graphics, Image Processing*, 31:2–18, 1985. (Cited on page 21.)
- [119] C. Mei and P. Rives. Calibration between a central catadioptric camera and a laser range finder for robotic applications. pages 532–537. IEEE. ISBN 0-7803-9505-0. doi: 10.1109/ROBOT.2006.1641765. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1641765>. (Cited on page 30.)
- [120] Vicente Milanés, David F. Llorca, Blas M. Vinagre, Carlos González, and Miguel A. Sotelo. Clavileño: Evolution of an autonomous car. *In Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, page 1129–1134, 2010. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5624971. (Cited on page 4.)
- [121] R. Mohan, G. Medioni, and R. Nevatia. Stereo error detection, correction, and evaluation. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-11:113–120, 1989. URL <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=16708&navigation=1>. (Cited on page 27.)
- [122] G. Monteiro, C. Premebida, P. Peixoto, and U. Nunes. Tracking and classification of dynamic obstacles using laser range finder and vision. *In in Workshop on "Safe Navigation in Open and Dynamic Environments -Autonomous Systems versus Driving Assistance Systems" at the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)*, 1, 2006. (Cited on page 43.)
- [123] H. P. Moravec. Towards automatic visual obstacle avoidance. *Proc. 5th Int. Joint Conf. Artificial Intell.*, 1:584, 1977. (Cited on page 21.)

- [124] Jorge J. Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis*, page 105–116. Springer, 1978. URL <http://link.springer.com/content/pdf/10.1007/BFb0067700.pdf>. (Cited on page 148.)
- [125] F. Moreno-Noguer, V. Lepetit, and P. Fua. Accurate non-iterative $O(n)$ solution to the PnP problem. In *IEEE 11th International Conference on Computer Vision, 2007. ICCV 2007*, pages 1–8. IEEE, October 2007. ISBN 978-1-4244-1630-1. doi: 10.1109/ICCV.2007.4409116. (Cited on page 31.)
- [126] Shigeyuki Morita. *Geometry of differential forms*. American Mathematical Soc., 2001. ISBN 9780821810453. (Cited on page 56.)
- [127] D.M. Mount and N.S. Netanyahu. Computationally efficient algorithms for high-dimensional robust estimators. *CVGIP: Graphical Models and Image Processing*, 56(4):289–303, July 1994. ISSN 1049-9652. doi: 10.1006/cgip.1994.1026. URL <http://www.sciencedirect.com/science/article/pii/S1049965284710261>. (Cited on page 141.)
- [128] Daniel Munoz, James Andrew Bagnell, and Martial Hebert. Co-inference for multi-modal scene analysis. In *Computer Vision—ECCV 2012*, page 668–681. Springer, 2012. URL http://link.springer.com/chapter/10.1007/978-3-642-33783-3_48. (Cited on page 43.)
- [129] Noboru Murata, Shuji Yoshizawa, and Shun-ichi Amari. Network information criterion-determining the number of hidden units for an artificial neural network model. *Neural Networks, IEEE Transactions on*, 5(6):865–872, 1994. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=329683. (Cited on page 168.)
- [130] V. S. Nalwa and T. O. Binford. On detecting edges. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-8:699–714, 1986. URL <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4767852&navigation=1>. (Cited on page 213.)
- [131] R. Nevatia and K. Babu. Linear feature extraction and description. *Comput. Graphics, Image Processings*, 13:257–269, 1980. (Cited on page 22.)
- [132] M.S. Nikulin. *Efficiency of a statistical procedure*. Encyclopedia of Mathematics. Springer, 2001. ISBN 978-1-55608-010-4. URL http://www.encyclopediaofmath.org/index.php/Efficiency_of_a_statistical_procedure. (Cited on page 146.)
- [133] A. Nuechter, O. Wulf, K. Lingemann, J. Hertzberg, B. Wagner, and H. Surmann. 3D mapping with semantic knowledge. In *RoboCup International Symposium*, 1, 2004. (Cited on page 42.)

- [134] D. Oberkamp, D. F DeMenthon, and L. S Davis. Iterative pose estimation using coplanar points. In , 1993 *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1993. Proceedings CVPR '93*, pages 626–627. IEEE, June 1993. ISBN 0-8186-3880-X. doi: 10.1109/CVPR.1993.341055. (Cited on page 30.)
- [135] Y. Ohta and T. Kanade. Stereo by intra- and inter-scanline search. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-7:139–154, 1985. URL <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4767639&navigation=1>. (Cited on page 39.)
- [136] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, 2001. (Cited on page 45.)
- [137] Mahamed GH Omran, Andries Petrus Engelbrecht, and Ayed Salman. Differential evolution methods for unsupervised image classification. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 2, page 966–973, 2005. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1554795. (Cited on page 191.)
- [138] Thomas J. Osgood and Yingping Huang. Sensor fusion calibration for driver assistance systems. In *2011 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pages 187–192. IEEE, July 2011. ISBN 978-1-4577-0576-2. doi: 10.1109/ICVES.2011.5983812. (Cited on page 65.)
- [139] Thomas J. Osgood and Yingping Huang. Calibration of laser scanner and camera fusion system for intelligent vehicles using Nelder–Mead optimization. *Measurement Science and Technology*, 24(3):035101, March 2013. ISSN 0957-0233, 1361-6501. doi: 10.1088/0957-0233/24/3/035101. URL <http://iopscience.iop.org/0957-0233/24/3/035101>. (Cited on pages 17, 49, and 209.)
- [140] Thomas J. Osgood, Yingping Huang, and K. Young. Minimisation of alignment error between a camera and a laser range finder using nelder-mead simplex direct search. In *2010 IEEE Intelligent Vehicles Symposium (IV)*, pages 779–786. IEEE, June 2010. ISBN 978-1-4244-7866-8. doi: 10.1109/IVS.2010.5548126. (Cited on pages 48, 49, and 60.)
- [141] Dan Pelleg and Andrew W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, page 727–734, 2000. URL http://pdf.aminer.org/000/335/443/x_means_extending_k_means_with_efficient_estimation_of_the.pdf. (Cited on page 38.)
- [142] J. Platt. Probabilistic outputs for support vector machines and comparison to regularize likelihood methods. *Advances in Large Margin Classifiers*, 1:61–74, 2000. (Cited on page 171.)

- [143] S. B. Pollard, J. E. W. Mayhew, and J. P. Frisby. PMF: a stereo correspondence algorithm using a disparity gradient limit. *Perception*, 14:449–470, 1981. (Cited on page 38.)
- [144] J. Ponce, M. Hebert, C. Schmid, and A. Zisserman. Toward category-level object recognition. *Lecture Notes in Computer Science*, 4170, 2007. (Cited on page 43.)
- [145] A. R. Pope. Model-based object recognition - a survey of recent research. *Technical Report TR-94-04*, 1, 1994. (Cited on page 43.)
- [146] T. J. Popham. *Tracking 3D Surfaces Using Multiple Cameras: A Probabilistic Approach*. PhD thesis, University of Warwick, 2010. URL <http://eprints.dcs.warwick.ac.uk/564/>. (Cited on page 23.)
- [147] I. Posner, D. Schröter, and P. Newman. Describing composite urban workspaces. In *Proc. of the Int. Conference on Robotics and Automation (ICRA)*, 1, 2007. (Cited on page 42.)
- [148] Ingmar Posner, Derik Schröeter, and Paul Newman. Online generation of scene descriptions in urban environments. *Robot. Auton. Syst.*, 56(11):901–914, November 2008. ISSN 0921-8890. doi: 10.1016/j.robot.2008.08.009. URL <http://dx.doi.org/10.1016/j.robot.2008.08.009>. (Cited on pages xi, 43, 44, 146, 160, and 187.)
- [149] Lutz Prechelt. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 11(4):761–767, 1998. URL <http://www.sciencedirect.com/science/article/pii/S0893608098000100>. (Cited on page 173.)
- [150] Edoardo Provenzi, Carlo Gatta, Massimo Fierro, and Alessandro Rizzi. A spatially variant white-patch and gray-world method for color image enhancement driven by local contrast. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(10):1757–1770, 2008. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4407718. (Cited on page 103.)
- [151] L. Quan and Z. Lan. Linear n-point camera pose determination. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(8):774–780, 1999. (Cited on page 30.)
- [152] Ian Read and Stephen Cox. Automatic pitch accent prediction for text-to-speech synthesis. In *INTERSPEECH*, page 482–485, 2007. URL http://www.uea.ac.uk/polopoly_fs/1.77246!/read-cox-interspeech-07.pdf. (Cited on page 176.)
- [153] Alessandro Rizzi, Carlo Gatta, and Daniele Marini. Color correction between gray world and white patch. *Electronic Imaging*, page 20–25, 2002. URL http://www.dti.unimi.it/~rizzi/papers/4662_44.pdf. (Cited on page 103.)

- [154] G. Robinson. Edge detection by compass gradient mask. *Comput. Graphics Image Processing*, 6:492–572, 1977. (Cited on page 38.)
- [155] Bryan C. Russell, Antonio Torralba, Kevin P. Murphy, and William T. Freeman. LabelMe: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1-3):157–173, 2008. URL <http://link.springer.com/article/10.1007/s11263-007-0090-8>. (Cited on page 46.)
- [156] D. Scaramuzza, A. Harati, and R. Siegwart. Extrinsic self calibration of a camera and a 3D laser range finder from natural scenes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007. IROS 2007*, pages 4164–4169. IEEE, October 2007. ISBN 978-1-4244-0912-9. doi: 10.1109/IROS.2007.4399276. (Cited on page 106.)
- [157] F. Schroff, A. Criminisi, and A. Zisserman. Object class segmentation using random forests. In *BMVC*, 1, 2008. (Cited on page 40.)
- [158] D. Schröter, M. Beetz, and J.-S. Gutmann. RG mapping: Learning compact and structured 2D line maps of indoor environments. In *Proc. of 11th IEEE ROMAN Conf.*, 1, 2002. (Cited on page 41.)
- [159] G. Schweighofer and A. Pinz. Robust pose estimation from a planar target. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2024–2030, December 2006. ISSN 0162-8828. doi: 10.1109/TPAMI.2006.252. (Cited on page 31.)
- [160] G. A. F. Seber and C. J. Wild. *Nonlinear regression*. 2003. Wiley-Interscience. (Cited on page 150.)
- [161] George A. F. Seber. *Multivariate observations*, volume 252. Wiley. com, 2009. URL http://books.google.co.uk/books?hl=en&lr=&id=UWk6YmtkhLgC&oi=fnd&pg=PP2&dq=Multivariate+Observations&ots=eakjDSvwXA&sig=I10g-Bp4B_oAQ3H4QA5iqARQ5qk. (Cited on pages 37 and 150.)
- [162] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, 1, 2008. (Cited on page 40.)
- [163] Ben Kröse Smagt and Patrick Van Der. An introduction to neural networks, 1996. URL <http://www.learnartificialneuralnetworks.com/>. (Cited on pages 31, 32, and 33.)
- [164] Xianfeng Song, Zheng Duan, and Xiaoguang Jiang. Comparison of artificial neural networks and support vector machine classifiers for land cover classification in northern china using a SPOT-5 HRG image. *International Journal of Remote Sensing*, 33(10):3301–3320, 2012. ISSN 0143-1161. doi: 10.1080/01431161.2011.568531. URL <http://www.tandfonline.com/doi/abs/10.1080/01431161.2011.568531>. (Cited on pages 159 and 218.)

- [165] Helmuth Spath. *The cluster dissection and analysis theory fortran programs examples*. Prentice-Hall, Inc., 1985. URL <http://dl.acm.org/citation.cfm?id=537092>. (Cited on page 37.)
- [166] James O. Street, Raymond J. Carroll, and David Ruppert. A note on computing robust regression estimates via iteratively reweighted least squares. *The American Statistician*, 42(2):152–154, 1988. URL <http://amstat.tandfonline.com/doi/abs/10.1080/00031305.1988.10475548>. (Cited on page 145.)
- [167] Jian Sun, Yin Li, Sing Bing Kang, and Heung-Yeung Shum. Symmetric stereo matching for occlusion handling. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, page 399–406, 2005. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1467470. (Cited on page 25.)
- [168] Shamik Sural, Gang Qian, and Sakti Pramanik. Segmentation and histogram generation using the HSV color space for image retrieval. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 2, page II–589, 2002. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1040019. (Cited on page 123.)
- [169] M. A. Tahir, A. Bouridane, and F. Kurugollu. An FPGA based coprocessor for GLCM and haralick texture features and their application in prostate cancer classification. *Analog Integrated Circuits and Signal Processing*, 43(2):205–215, 2005. URL <http://link.springer.com/article/10.1007/s10470-005-6793-2>. (Cited on pages 135 and 216.)
- [170] Igor V. Tetko, David J. Livingstone, and Alexander I. Luik. Neural network studies. 1. comparison of overfitting and overtraining. *Journal of chemical information and computer sciences*, 35(5):826–833, 1995. URL <http://pubs.acs.org/doi/abs/10.1021/ci00027a006>. (Cited on pages 173 and 174.)
- [171] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, and Gabriel Hoffmann. Stanley: The robot that won the DARPA grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006. URL <http://onlinelibrary.wiley.com/doi/10.1002/rob.20147/abstract>. (Cited on page 4.)
- [172] Bin Tian, Mukhtiar A. Shaikh, Mahmood R. Azimi-Sadjadi, Thomas H. Vonder Haar, and Donald L. Reinke. A study of cloud classification with neural networks using spectral and textural features. *Neural Networks, IEEE Transactions on*, 10(1):138–151, 1999. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=737500. (Cited on page 135.)
- [173] Gui Yun Tian and Rong-Sheng Lu. Hybrid vision system for online measurement of surface roughness. *JOSA A*, 23(12):3072–3079, 2006. URL <http://www.opticsinfobase.org/abstract.cfm?id=117917>. (Cited on page 135.)

- [174] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001. URL <http://onlinelibrary.wiley.com/doi/10.1111/1467-9868.00293/abstract>. (Cited on page 38.)
- [175] Chuen-Lin Tien, You-Ru Lyu, and Shiao-Shan Jyu. Surface flatness of optical thin films evaluated by gray level co-occurrence matrix and entropy. *Applied Surface Science*, 254(15):4762–4767, 2008. URL <http://www.sciencedirect.com/science/article/pii/S0169433208001475>. (Cited on page 135.)
- [176] K. Tieu and P. Viola. Boosting image retrieval. In *CVPR*, 1, 2000. (Cited on page 45.)
- [177] Joseph Tighe and Svetlana Lazebnik. Superparsing: scalable nonparametric image parsing with superpixels. In *Computer Vision—ECCV 2010*, page 352–365. Springer, 2010. URL http://link.springer.com/chapter/10.1007/978-3-642-15555-0_26. (Cited on page 46.)
- [178] Department for Transport. *Reported casualties by severity, road user type and country, United Kingdom, latest available year - RAS30034*. DfT STATS19, PSNI, September 2011. URL <http://www.dft.gov.uk/statistics/tables/ras30034/>. Table ID RAS30034. (Cited on page 1.)
- [179] C. J. Van Rijsbergen. Information retrieval. dept. of computer science, university of glasgow. URL: citeseer.ist.psu.edu/vanrijsbergen79information.html, 1979. (Cited on page 176.)
- [180] Various. PreVent IP, 2009. URL <http://www.prevent-ip.org/>. (Cited on page 2.)
- [181] Paul F. Velleman and David C. Hoaglin. *Applications, basics, and computing of exploratory data analysis*, volume 142. Duxbury Press Boston, 1981. URL http://www.library.cornell.edu/dlit/presentations/IFUP_Catalog.pdf. (Cited on pages 152 and 153.)
- [182] Kiri Wagstaff and Claire Cardie. Clustering with instance-level constraints. In *AAAI/IAAI*, page 1097, 2000. URL <http://www.aaai.org/Papers/AAAI/2000/AAAI00-180.pdf>. (Cited on page 37.)
- [183] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In *ICML*, volume 1, page 577–584, 2001. URL <https://web.cse.msu.edu/~cse802/notes/ConstrainedKmeans.pdf>. (Cited on page 37.)
- [184] S. Wasielewski and O. Strauss. Calibration of a multi-sensor system laser rangefinder/camera. In *Intelligent Vehicles '95 Symposium., Proceedings of the*, pages 472–477. IEEE, September 1995. ISBN 0-7803-2983-X. doi: 10.1109/IVS.1995.528327. (Cited on page 30.)

- [185] Jason Weston and Chris Watkins. Multi-class support vector machines. Technical report, Citeseer, 1998. URL <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.50.9594&rep=rep1&type=pdf>. (Cited on page 168.)
- [186] C. Wojek and B. Schiele. A dynamic CRF model for joint labeling of object and scene classes. In *ECCV*, 1, 2008. (Cited on page 40.)
- [187] Oliver Woodford, Philip Torr, Ian Reid, and Andrew Fitzgibbon. Global stereo reconstruction under second-order smoothness priors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(12):2115–2128, 2009. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5128908. (Cited on page 26.)
- [188] C. Wulf, O. Brenneke, and B. Wagner. Colored 2D maps for robot navigation with 3D sensor data. In *In Proc. of IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)*, 1, 2004. (Cited on page 216.)
- [189] Xuehan Xiong and Daniel Huber. Using context to create semantic 3D models of indoor environments. In *BMVC*, page 1–11, 2010. URL <http://www.bmva.org/bmvc/2010/conference/paper45/paper45.pdf>. (Cited on page 42.)
- [190] Xuehan Xiong, Daniel Munoz, J. Andrew Bagnell, and Martial Hebert. 3-d scene analysis via sequenced predictions over points and regions. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, page 2609–2616, 2011. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5980125. (Cited on page 42.)
- [191] Yongzhu Xiong, Zhengdong Zhang, and Feng Chen. Comparison of artificial neural network and support vector machine methods for urban land use/cover classifications from remote sensing images a case study of guangzhou, south china. In *2010 International Conference on Computer Application and System Modeling (ICCASM)*, volume 13, pages V13–52–V13–56, 2010. doi: 10.1109/ICCASM.2010.5622651. (Cited on pages 159 and 218.)
- [192] Hongming Zhang, Wen Gao, Xilin Chen, and Debin Zhao. Object detection using spatial histogram features. *Image and Vision Computing*, 24(4): 327–341, 2006. URL <http://www.sciencedirect.com/science/article/pii/S0262885605002088>. (Cited on page 216.)
- [193] Qilong Zhang and R. Pless. Extrinsic calibration of a camera and laser range finder (improves camera calibration). In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, pages 2301–2306, Sendai, Japan. doi: 10.1109/IROS.2004.1389752. (Cited on page 30.)

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst’s seminal book on typography “*The Elements of Typographic Style*” [21]. `classicthesis` is available for both \LaTeX and \LyX :

<http://code.google.com/p/classicthesis/>

Final Version as of 31st January 2014 (`classicthesis` version 2.1).

